Math

Report No. 480

ON THE SYNTHESIS BY INTEGER PROGRAMMING OF OPTIMAL NOR GATE NETWORKS FOR FOUR VARIABLE SWITCHING FUNCTIONS

by

Jay Niel Culliney

September 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 480

ON THE SYNTHESIS BY INTEGER PROGRAMMING OF OPTIMAL NOR GATE NETWORKS FOR FOUR VARIABLE SWITCHING FUNCTIONS*

by

Jay Niel Culliney

September 1971

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* This work was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois, September 1971 and supported in part by the National Science Foundation under Grant No. NSF GJ-503.

ACKNOWLEDGMENT

The author would like to express his deep gratitude to his advisor, Professor Saburo Muroga, for his enthusiastic guidance and encouragement during the preparation of this thesis, and also for his careful reading and valuable suggestions for the improvement of the original manuscript.

This work was supported in part by the National Science Foundation under Grant No. NSF GJ-503.

This paper is dedicated to two friends, Janet Lynn and John Misha Petkevich.

TABLE OF CONTENTS

1. INTRODUCTION

This report represents an effort to apply the implicit enumeration algorithm developed in [2, 14, 15] to the problem of synthesizing optimal networks for four variable switching functions. In particular, NOR gates were used in the synthesis of these networks; complements of variables were assumed to be unavailable.

To avoid superfluous efforts, the 65,536 functions of four variables were divided into 3,984 P (permutation of variables) equivalence classes. From among the representative functions of these classes, were selected 438 functions for which optimal networks were synthesized. These selected functions required from 3 to 8 NOR gates in their network realizations (networks, and the corresponding functions, of 2 or fewer NOR gates are easily exhausted by hand*).

Although networks with the minimal numbers of gates were already known for 436 of these functions[12], this report is believed to be the first listing of networks for these functions where the sum of the number of connections of external variables to gates plus the number of interconnections among gates is also known to be minimal.

---

* See Appendix D.

2. INTEGER PROGRAMMING AND IMPLICIT ENUMERATION

The problem of finding an optimal network to realize a given function is conveniently posed as an integer programming problem. This approach proves to be very flexible, accommodating a wide spectrum of optimality criteria.

The general integer programming problem with N unknown variables and m constraints may be stated:

$$\begin{aligned} &\text{minimize} && \vec{c}\,\vec{x} \\ &\text{subject to} && \vec{b} + A\,\vec{x} \geq 0 \\ &\text{where} && \vec{c} = [c_1, \ldots, c_N],\ c_i \geq 0 \qquad i = 1, \ldots, N; \\ & && \vec{b} = [b_1, \ldots, b_m]; \\ &\text{and} && A = [a_{ij}]_{mN} \end{aligned}$$

are coefficient matrices and $\vec{x} = [x_1, \ldots, x_N]$ is an N-dimensional vector of variables. In this work, all coefficients and variables are integral; and the variables, x_i's, assume only the values 1 or 0.

The objective function, $\vec{c}\,\vec{x}$, represents the "cost" of a proposed solution $\vec{x}$. In minimizing $\vec{c}\,\vec{x}$, we are finding the $\vec{x}$ which yields the lowest "cost". One way to do this is to generate every possible $\vec{x}$ (there are 2^N such vectors since $x_i = 0, 1$), test each to see if it satisfies the constraints $\vec{b} + A\,\vec{x} \geq 0$, discard those that fail, and select the $\vec{x}$'s (or $\vec{x}$, if there is only one) with the lowest cost out of those remaining. These $\vec{x}$'s must then be optimal solutions to the integer programming problem by virtue of the fact that all potential solutions would have been explicitly enumerated and examined.

This algorithm is theoretically possible, but, for large values of N, it becomes impossible in practice. For example, for two problems in this work, 1112 variables are employed. To examine 2^{1112} possible $\vec{x}$'s individually, say at the rate of one billion $\vec{x}$'s per second, would require about 2^{1057} years (plus or minus a couple orders of magnitude).

Luckily, a vastly more efficient algorithm is available for solving this type of zero-one problem: the implicit enumeration method.* As its name suggests, it enumerates implicitly all of the 2^N solutions and selects the best feasible solution(s).

To illustrate how constraints may be written as inequalities, consider a person making payments on his monthly bills. (For the sake of this example, only allow the x_i to assume integral values other than 0 or 1.) One of the constraints this person must work under is:

$$-\sum_i (\text{PAYMENT ON BILL } i) + \sum_k (\text{LOAN FROM COMPANY } k) + \text{MONTHLY INCOME} \geq 0.$$

In this case, the monthly income is an element of $\vec{b}$, the coefficients of the corresponding row of A are - 1's and + 1's (and 0's for columns corresponding to any variables not appearing in this constraint), and the payments on various bills and loans from various companies are elements of $\vec{x}$.

To understand the implicit enumeration scheme, several definitions are necessary:

---

* For details of the implicit enumeration algorithm, see [2], [8], and [11]. The mathematical theory behind this algorithm is outlined in [1], [7], [8], and [9].

A solution is an assignment of the values 0 or 1 to all the variables of $\vec{x}$.

A solution which satisfies $\vec{b} + A\vec{x} \geq 0$ is said to be feasible; a solution which does not, is said to be infeasible.

A feasible solution which minimizes $\vec{c}\,\vec{x}$ is called an optimal feasible solution.

A partial solution, S,is defined as an ordered assignment of binary values to a proper subset of the N variables of $\vec{x}$.

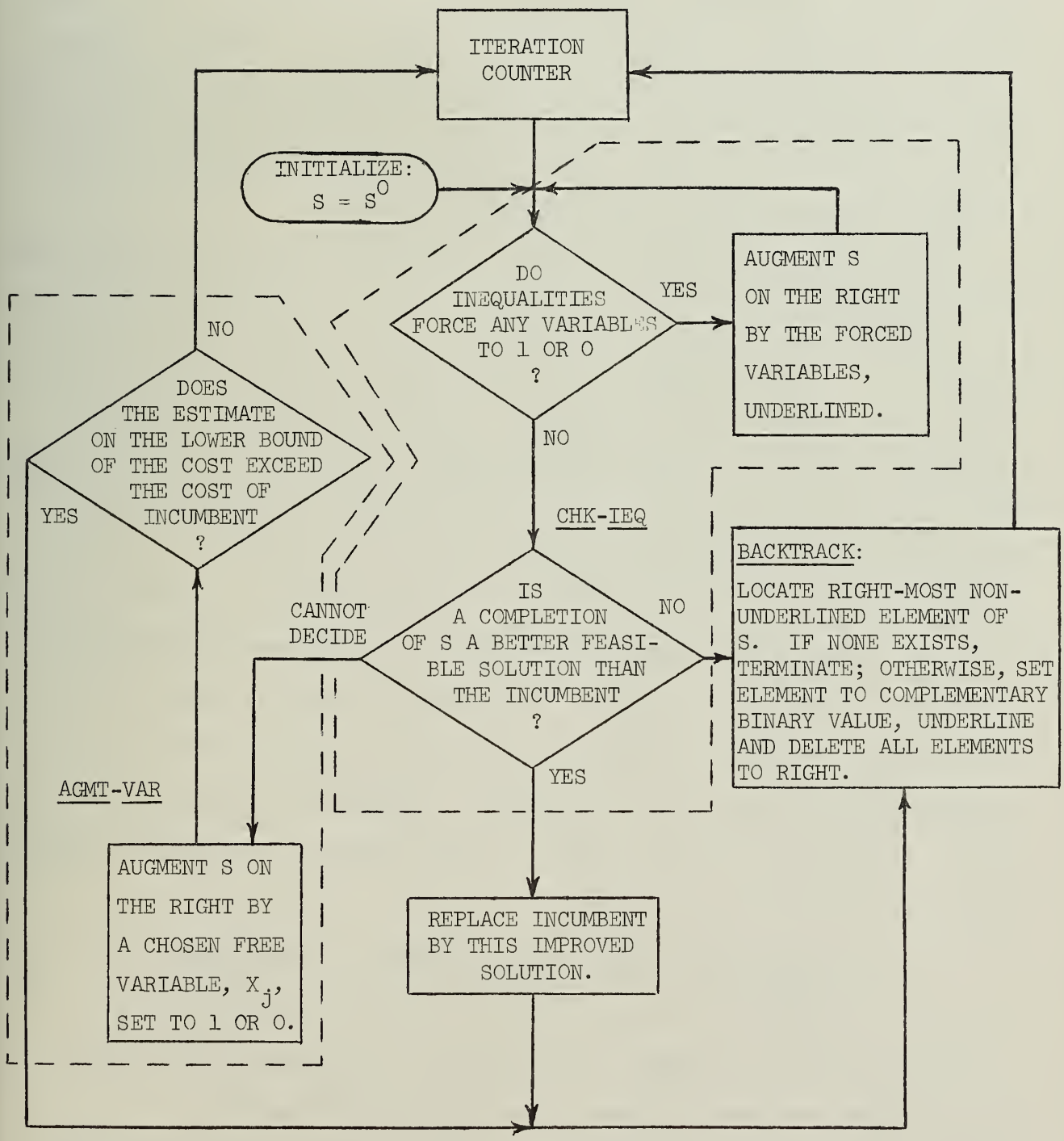
A variable that has not been assigned a value is called a free variable; a variable that has been assigned is called a fixed variable.

A completion of a partial solution S is a solution consisting of S followed by a binary assignment to all free variables of S.

Since the implicit enumeration algorithm is discussed in depth elsewhere[13] and since many of the details are unnecessary for the purpose of understanding this report, the algorithm will only be outlined as shown in the simplified flowchart of Fig. 2.1. (However, the part of the algorithm which chooses variables to augment the partial solutions, being tailored specifically for this NOR network problem, will be discussed in more detail in Section 4.)

Starting with a given partial solution S^0 (which may be an assignment to no variables at all) and an incumbent solution (the feasible solution having the smallest value of the objective function obtained so far), the main block, labeled "CHK-IEQ" (for "check inequalities"), is entered. In a typical application of the implicit enumeration algorithm to the problem of finding an optimal network for a function, the incumbent might represent the

Fig. 2.1 Implicit enumeration algorithm



best network found by using a heuristic method. Or, if no solution is known at all, the incumbent might just be a "dummy" with a sufficiently high objective function value to guarantee that there is some real solution(s) (with a lower value of the objective function) which will displace it. (If one's purpose is to show the infeasibility of the integer programming problem, the objective function value of the initial "dummy" incumbent must be set higher than the objective function values of all potential solutions.)

Upon entering "CHK-IEQ", we examine inequalities and may find that certain free variables must be set (i.e., forced) to 0 or 1 in order that all of the inequalities can be satisfied. The inequalities are scanned repeatedly until no more free variables may be assigned. The original partial solution, S^0, augmented by the free variables thus assigned, becomes a new partial solution, S^1. This new partial solution, S^1, is then checked to determine which of the following three cases has occurred:

(1) A feasible solution has been found. The completion of S^1 obtained by setting all free variables to 0 is a feasible solution. Compare it with the incumbent solution and retain the better (i.e., the one with the lower objective function value) as the incumbent. Initiate the backtrack procedure to obtain a new partial solution, S^2 (see Appendix A for a brief explanation of the backtrack procedure).

(2) All completions of the partial solution are infeasible. If at least one inequality is not satisfied by S^1, whatever binary values are assigned to the free variables, then discard S^1 immediately by initiating the backtrack procedure to obtain a new partial solution, S^2.

(3) Augment the partial solution. If neither case (1) nor (2) occurs, assign a free variable to the value 1, forming S^2. The choice of this

variable is extremely important to the convergence rate of the algorithm and should take into consideration the type of problem being solved.

After obtaining S^2, reenter the block "CHK-IEQ" and repeat the process.

Cycling through this process until the algorithm terminates (this will occur when we try to backtrack and find all fixed variables underlined) will result in the implicit enumeration of all possible solutions. Also, the optimum solution will be the incumbent at the termination of the algorithm.

Without knowing more details of the algorithm, it might appear difficult (i.e., time-consuming) to choose among the above cases during each cycle. In fact, this can be done rather quickly as explained in [11].

The algorithm will, of course, converge in a finite number of steps, but its efficiency will depend heavily on the nature of the individual problem being solved. Computational experience shows the tailoring of the "AGMT-VAR" block in Fig. 2.1 (the subroutine which augments the partial solution when case (3) occurs) to a particular problem increases the rate of convergence.[11]

3. ALL-INTERCONNECTION NOR SYNTHESIS

At least one earlier paper[18] presented optimal combinational networks of gates, based on the integer programming approach discussed in [14], [15]. This synthesis was based upon the feed-forward approach. In this approach the NOR gates were prearranged according to levels such that each gate could only receive inputs from the external variables x_1, x_2, ..., x_n or from the outputs of gates of the preceding levels (this was done by simply not providing variables to represent any other possible interconnections among the gates). This restriction forces every solution to be of the desired "feed-forward" type.

Later papers[2], [5] employ a somewhat different formulation to obtain the same "feed-forward" type networks. This new formulation, the all-interconnection approach, allows interconnections between every distinct pair of gates (i.e., variables are provided to represent every possible interconnection). Then inequalities are added to prevent loops from occurring in the generated networks. This new formulation, although apparently slower (because more variables and inequalities are involved), actually speeds up the network synthesis considerably.

This paper, using the faster all-interconnection approach, presents computational results in the synthesis of single output, multilevel, loop-free networks of NOR gates which realize given 4-variable switching functions.

In determining the optimal networks presented in this paper, certain assumptions were made: complements of external variables were not available, infinite fan-in and fan-out were allowed, and the number of levels of logic was unrestricted. However, any of these constraints could have been easily applied by simple modifications of the program used to obtain the optimal networks.

3.1 Introduction of Notation

In order to explain how the optimization problem can be characterized by a corresponding integer programming problem, certain notations must be introduced. Assume that we are trying to characterize a network of R gates and n external variables (x_1, ..., x_n) to realize a function, f, of n (or in degenerative cases, fewer) variables.

First we need variables to represent all possible connections (of external variables to gates) and interconnections (of gates to other gates):

w_{ik} is the connection from the i-th external variable, x_i, to the k-th gate

$w_{ik} = 1$ will mean that the connection exists

$w_{ik} = 0$ will mean that the connection does not exist

α_{ik} is the interconnection from gate i to gate k

$\alpha_{ik} = 1$ will mean that the interconnection exists

$\alpha_{ik} = 0$ will mean that the interconnection does not exist

In addition to these variables, others must be introduced to allow the representation of the gate outputs for all 2^n possible external input vectors. (The input vectors, (x_1, x_2, x_3, x_4), (using $n = 4$ as an example) are assumed to be ordered as follows: (0, 0, 0, 0) is the 1-st, (0, 0, 0, 1) is the 2-nd, (0, 0, 1, 0) is the 3-rd, ..., (1, 1, 1, 1) is the 2^4-th.)

$P_i^{(j)}$ is the output of gate i corresponding to the j-th input vector.

But in order to determine the output of each gate in a network, we must know the inputs to that gate. Inputs to a gate are classified as two types:

external inputs, which are from external variables, and internal inputs, which are from outputs of other gates in the network. So an input to a gate is dependent on two things: whether or not a connection (interconnection) exists from a certain external variable (gate) and, if it does, the state (1 or 0) of that variable (gate output):

$w_{ik}\, x_i^{(j)}$ is the input to gate k from the i-th external variable for the j-th input vector

$\beta_{ik}^{(j)} = \alpha_{ik}\, P_i^{(j)}$ is the input to gate k from the i-th gate for the j-th input vector.

To simplify later discussions, let an input connection to a gate be defined as either an external variable connection or a gate interconnection which supplies an input to that gate.

3.2 Problem Formulation

In the last section, it was mentioned that we are explaining the problem formulation for R gates, i.e., the number of gates is fixed for a given formulation. This might raise the question of how the number of gates is minimized in the network for a given function, since we seemingly must know the optimal number of gates in advance in order to choose the correct formulation with which to run the program (to further optimize the network by minimizing the number of input connections). The answer is simple, of course. First try to solve the problem with the R = 1 formulation. If it is infeasible, repeat the attempt using the R = 2 formulation. Continue incrementing R and solving

the corresponding formulations until a feasible solution occurs for a certain value of R; this value must then represent the minimum number of gates necessary to realize the given function.

Of course, if a lower bound, r, is known on the number of gates needed to realize a given function, we can skip all attempts to realize the function with fewer than r gates (i.e., start with R = r rather than R = 1). In fact, this is what has been done for the functions whose networks were optimized in this paper. Based on previous research of N. Ikeno, A. Hashimoto, and K. Naito[12], the optimal number of gates for almost every function was known in advance; and, therefore, each function could be run with the correct formulation on the first attempt (thereby saving some computation time).

The following sections will explain explicitly the objective function and inequalities comprising the integer programming problem. It is again pointed out that this formulation is for a network of R gates and n external variables under the assumptions stated above in the introduction to Section 3.

3.3 Objective Function

Recall that we want to minimize the number of input connections used to realize a given function, f, for a fixed number of gates, R. So the objective function to be minimized is simply,

$$\Sigma \text{ CONNECTIONS} + \Sigma \text{ INTERCONNECTIONS, or}$$

$$\sum_{k=1}^{R} \left[\sum_{i=1}^{n} w_{ik} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \alpha_{\ell k} \right] . \qquad (1)$$

3.4 Inequalities Describing Gates

This section includes both the basic inequalities needed to describe the gates and supplementary inequalities to speed the computation or to restrict the network.

3.4.1 Regular Inequalities

These inequalities describe the function of a NOR gate for each gate in the network. They are just a mathematical expression of the fact that the output of a NOR gate must be a 1 if all its inputs are 0, or the output must be a 0 if at least one of its inputs is a 1.

These inequalities are written in two sets: the first set is for non-output gates whose outputs change dynamically during the computation, the second set is for the output gate of the network (designated gate number 1) whose output is fixed during the computation.*

For all gates in the network except the output gate (k = 2, ..., R):

$$\sum_{i=1}^{n} w_{ik}\, x_i^{(j)} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \beta_{\ell k}^{(j)} \geq 1-U\ P_k^{(j)} \tag{2}$$

$$\sum_{i=1}^{n} w_{ik}\, x_i^{(j)} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \beta_{\ell k}^{(j)} \leq U(1-P_k^{(j)}) \tag{3}$$

$$j = 1, \ldots, 2^n.$$

---

* The actual implementation used was somewhat different, but it was the same in effect.

And for the output gate (k = 1):

$$\sum_{i=1}^{n} w_{ik}\, x_i^{(j)} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \beta_{\ell k}^{(j)} \geq 1 \qquad \text{if } f(\vec{x}^{(j)}) = 0 \qquad (4)$$

$$\sum_{i=1}^{n} w_{ik}\, x_i^{(j)} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \beta_{\ell k}^{(j)} \leq 0 \qquad \text{if } f(\vec{x}^{(j)}) = 1 \qquad (5)$$

$$j = 1, \ldots, 2^n.$$

The value U in the above inequalities (2) and (3) is a sufficiently large positive number such that (2) becomes non-restrictive when $P_k^{(j)} = 1$ and (3) becomes non-restrictive when $P_k^{(j)} = 0$ (i.e., $U \geq R + n - 1$).

Notice that each of the above gate description inequalities represents 2^n inequalities - one for every possible input vector.

So that the inequalities (2) - (5) could be linear (note that the $w_{ik}\, x_i^{(j)}$ terms are linear since $\vec{x}^{(j)}$ is known for every j), the "beta" terms were defined earlier, $\beta_{ik}^{(j)} = \alpha_{ik} P_i^{(j)}$, where $\beta_{ik}^{(j)}$ represents the input to the k-th gate from the i-th gate for the j-th input vector $\vec{x}^{(j)}$. At first it may appear that one set of non-linear inequalities have just been replaced by another, but, in fact, for each j the relation $\beta_{ik}^{(j)} = \alpha_{ik} P_i^{(j)}$ can be written as two linear inequalities:

$$-1 \leq -P_i^{(j)} - \alpha_{ik}^{(j)} + \beta_{ik}^{(j)} \qquad (6)$$

$$0 \leq P_i^{(j)} + \alpha_{ik}^{(j)} - 2\beta_{ik}^{(j)} \qquad (7)$$

for $j = 1, \ldots, 2^n$; $i = 1, \ldots, R$; $k = 1, \ldots, R$; and $i \neq k$.

3.4.2 Additional Inequalities

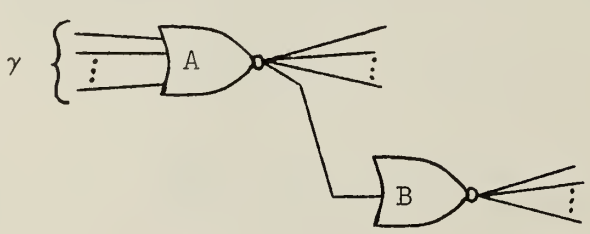
The preceding inequalities are sufficient to describe a network of R NOR gates with n external variables. These are referred to as regular inequalities. But to improve the convergence rate of the implicit enumeration algorithm, other inequalities, referred to as additional inequalities, are also used.

3.4.2.1 Input Conditions

Each NOR gate (except the output gate) must have at least one input connection from the set of external variables, or at least two input connections from other gates. (If fan-in, fan-out restrictions will be imposed, change the following inequalities to require at least one input connection from the external variables or at least one from other gates.)

Obviously, for a gate to function as a part of a network it must have at least one input connection, but the reason for requiring at least two interconnections from other gates (in the absence of any connections from external variables) may not be so immediately clear.

Suppose there is a gate B (not the output gate) that is fed only from one other gate (A). The general case is shown:



If such a subnetwork existed, we could reduce the number of gates needed for the entire network by eliminating gate B and connecting the set of inputs to A, γ, to every gate formerly fed by B. Hence, no such gate B can exist in an optimal network.

The necessary inequalities are:

$$2 \sum_{i=1}^{n} w_{ik} + \sum_{\substack{\ell=1 \\ \ell \neq k}}^{R} \alpha_{\ell k} \geq 2 \qquad k = 2, \ldots, R \qquad (8)$$

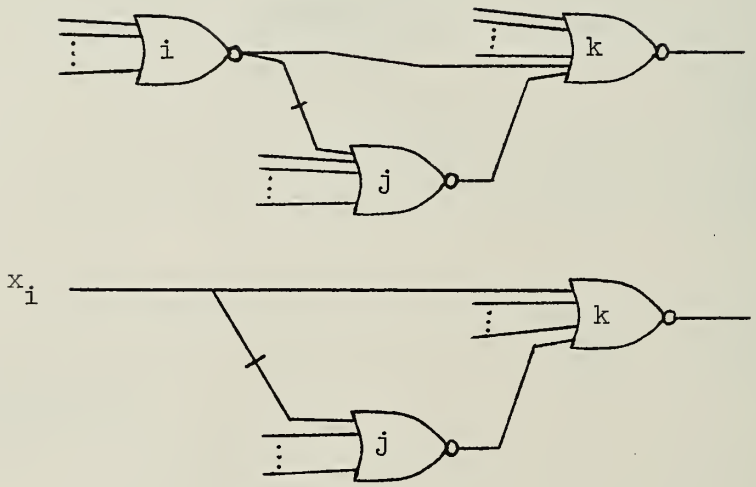
3.4.2.2 Output Conditions

Each NOR gate (except the output gate) must have at least one output to another gate in the network. This obvious requirement is expressed as:

$$1 \leq \sum_{\substack{k=1 \\ i \neq k}}^{R} \alpha_{ik} \qquad i = 2, \ldots, R \qquad (9)$$

3.4.2.3 Triangular Conditions

The following triangular subnetworks cannot appear in an optimal circuit and may therefore be prohibited by inequalities. In both cases gate j is assumed to have only one output connection.



The redundant input connections are marked by a short bar. That these are in fact redundant can be easily seen by considering separately the case when the output of gate i (or input x_i) is 1 and the case when it is 0. Observe that when a gate output in a network of NOR gates is 0, the gate is effectively disconnected from the network.

The inequalities used to prohibit these configurations are respectively:

$$\alpha_{ij} + \alpha_{ik} + \alpha_{jk} \leq 2 + \sum_{\substack{\ell=1 \\ \ell \neq k \\ \ell \neq j}}^{R} \alpha_{j\ell} \qquad \begin{array}{l} i = 1, \ldots, R \\ j = 1, \ldots, R \\ k = 1, \ldots, R \\ i \neq j,\ i \neq k,\ j \neq k; \end{array} \qquad (10)$$

$$w_{ij} + w_{ik} + \alpha_{jk} \leq 2 + \sum_{\substack{\ell=1 \\ \ell \neq k \\ \ell \neq j}}^{R} \alpha_{j\ell} \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, R \\ k = 1, \ldots, R \\ j \neq k. \end{array} \qquad (11)$$

3.4.2.4 Generalized Triangular Conditions

Somewhat similar to the preceding redundant triangular subnetworks are configurations in which a gate output connects both to the output gate and to some other gate in the network. By again considering the cases when this gate output is 0 or 1, it is easily seen that every connection of this gate output to a non-output gate is redundant. Therefore, these configurations cannot occur in an optimal network.

The appropriate restrictive inequalities are:

$$U\,\alpha_{il} + \sum_{k=2}^{R} \alpha_{ik} \leq U \qquad i = 2, \ldots, R \qquad (12)$$

where U has the value chosen earlier.

3.4.3 Fan-In Fan-Out Restrictions

Although fan-in, fan-out restrictions were not applied in obtaining the optimal networks found here, the program used has the option of generating the inequalities to impose these restrictions. The form of these inequalities, for the case when 3 is the maximum fan-in, fan-out, is:

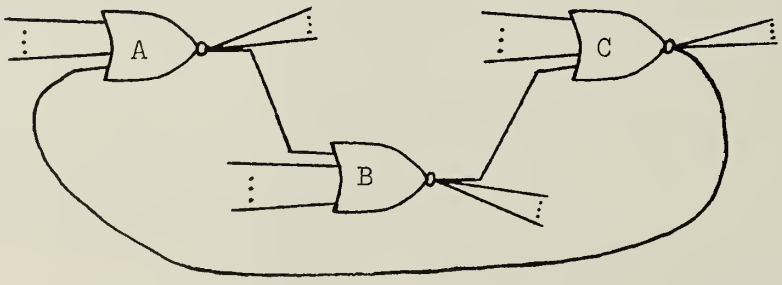
$$\text{(fan-in)} \qquad 3 \geq \sum_{\substack{k=1 \\ k \neq i}}^{R} \alpha_{ki} + \sum_{j=1}^{n} w_{ji} \qquad i = 1, \ldots, R \qquad (13)$$

$$\text{(fan-out)} \qquad 3 \geq \sum_{\substack{k=1 \\ i \neq k}}^{R} \alpha_{ik} \qquad i = 2, \ldots, R. \qquad (14)$$

3.4.4 Loop-Free Conditions

This last set of inequalities is a necessary part of the all-interconnection formulation. These inequalities insure that any networks obtained are of the feedforward type, i.e., no "feedback" loops will occur.

In this group of inequalities, there is one inequality for each possible "loop" of gates. For example, to prevent the subnetwork



we would use the inequality $\alpha_{AB} + \alpha_{BC} + \alpha_{CA} \leq 2$. Inequalities must be provided for loops consisting of from 2 to R - 1 gates. The output gate is

not included in any of these loops since, before the computation begins, all of the variables representing output connections from this gate to others in the network will be placed in the initial partial solution, set to 0, and underlined.

The number of these inequalities grows very rapidly compared to the other types as the number of gates increases.

The set of these inequalities may be expressed, somewhat awkwardly, as:

$$
\begin{aligned}
&\alpha_{ij} + \alpha_{ji} \leq 1 && i = 2, \ldots, R \\
& && j = 2, \ldots, R \quad i > j \\
&\alpha_{ij} + \alpha_{jk} + \alpha_{ki} \leq 2 && i = 2, \ldots, R \quad j = 2, \ldots, R \\
& && k = 2, \ldots, R \quad i > j, k \quad j \neq k \\
&\quad \cdot \quad \cdot \quad \cdot \\
&\alpha_{i_1 i_2} + \alpha_{i_2 i_3} + \ldots + \alpha_{i_{R-1} i_1} \leq R-2 && i_1 = 2, \ldots, R; \; i_2 = 2, \ldots, R; \\
& && \ldots; \; i_{R-1} = 2, \ldots, R; \\
& && i_1 > i_2, i_3, \ldots, i_{R-1}; \\
& && \text{all } i_k\text{'s distinct.}
\end{aligned}
\qquad (15)
$$

The total number of the loop-preventing inequalities is:

$$\sum_{i=2}^{R-1} \frac{(R-1)\,(R-2)\,\ldots\,(R-i)}{i}$$

4. AUGMENTATION OF PARTIAL SOLUTION

As shown in the simplified flow chart of the implicit enumeration algorithm, there are cases (in the CHK-IEQ block) when it is unknown whether or not there exist completions of the current partial solution with costs less than or equal to the cost of the incumbent. Having fixed, in an earlier part of CHK-IEQ, all of the free variables forced to certain values by the inequalities, the only direction in which to proceed is to choose some free x_j and set it to 0 or 1. Hopefully, this will cause enough variables to be set by the inequalities on the next iteration (each iteration corresponds to a pass through the "Iteration Counter" block in Fig. 2.1) to enable a determination of whether or not a better feasible solution than the incumbent exists as a completion of the current partial solution. If still more information is required, another variable is selected to be set to 0 or 1, and so forth.

The choice of the variables to be set in these cases is very important; consistently good choices improve computation time considerably. This choice takes place in the AGMT-VAR block of Fig. 2.1, which is detailed in Fig. 4.1.

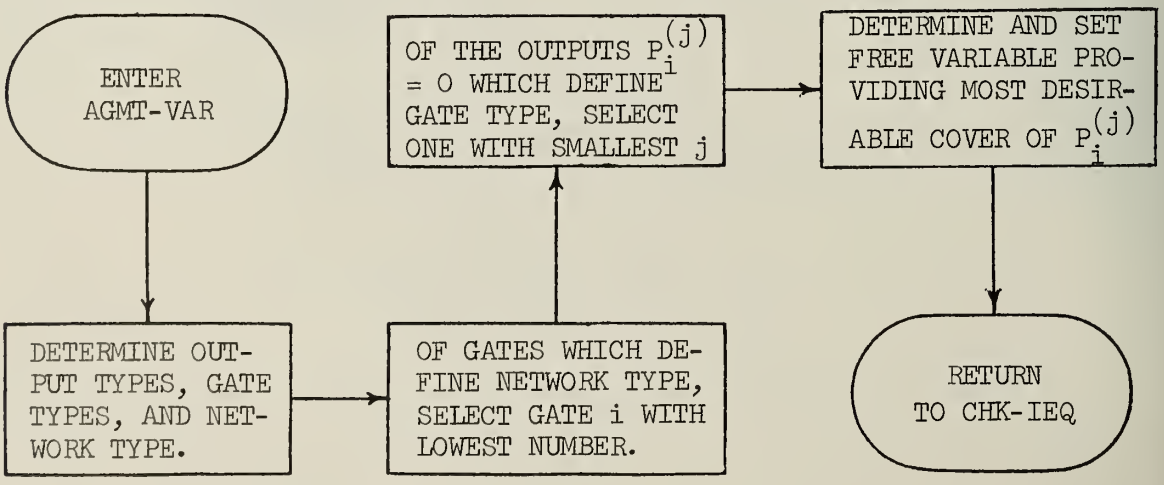


Fig. 4.1 AGMT-VAR detailed

4.1 Covers

Consider the output $P_k^{(j)}$ of gate k for the input vector $\vec{x}^{(j)}$. This variable is either 0, 1, or unassigned. If $P_k^{(j)}$ is unassigned, it makes no demands on the assignments of other variables; and, without considering other factors, there is no justification for making an assignment one way or the other (0 or 1). If $P_k^{(j)} = 1$, the inequalities (3) in Section 3.4.1 have already forced all of the influenced variables (w_{ik}'s and $\beta_{\ell k}^{(j)}$'s to 0. For the third case, $P_k^{(j)} = 0$, the inequalities (2) in Section 3.4.1 become restrictive and require at least one of the inputs to gate k to be a 1 for $\vec{x}^{(j)}$. If this condition is satisfied for the particular $P_k^{(j)} = 0$, then it is said to be covered, otherwise it is uncovered (remember that this discussion only applies to $P_k^{(j)}$'s with the assigned value 0). Thus, a cover of $P_k^{(j)}$ (= 0) is any source (either a gate or an external input variable) providing an input with the value 1 to gate k for $\vec{x}^{(j)}$.

Before proceeding further, certain definitions are necessary:

An isolated gate is a gate i for which no α_{ij} (j = 1, ..., R; j $\neq$ i) is 1 in the corresponding partial solution.

The classifications of possible covers of an output $P_k^{(j)} = 0$ are:

COV - if there exists an i or an ℓ such that $w_{ik}\, x_i^{(j)} = 1$ or $\beta_{\ell k}^{(j)} = 1$ (i.e., $P_k^{(j)}$ is COVered);

G^* - if there exists a gate ℓ such that $\alpha_{\ell k} = 1$ and $P_\ell^{(j)} = *$ ("*" will denote "unassigned") (G^* represents the type of the possible cover, gate ℓ);

VC^* - if there exists an external variable i such that $x_i^{(j)} = 1$ and $w_{ik} = *$ (VC^* represents the type of the possible cover, variable i);

GC* - if there exists a non-isolated gate ℓ such that $\alpha_{\ell k} = *$ and $P_\ell^{(j)} = 1$ (GC* represents the type of the possible cover, gate ℓ);

G*C* - if there exists a non-isolated gate ℓ such that $\alpha_{\ell k} = *$ and $P_\ell^{(j)} = *$ (G*C* represents the type of the possible cover, gate ℓ);

NWG - if there exists an isolated gate ℓ such that $\alpha_{\ell k} = *$ and $P_\ell^{(j)} = *$ (NWG represents the type of the possible cover, NeW Gate ℓ).

The above classifications are then ordered (in a list termed a desirability order) according to decreasing desirability:

COV, G*, VC*, GC*, G*C*, and NWG.

(Roughly, the more desirable a cover is, the less "disturbance" is caused to the network when the appropriate free variables are fixed in order to cover the $P_k^{(j)}$.)

4.2 Variable Augmentation Scheme

Using the above desirability order, output types, gate types, and network types may be defined.

For each output $P_k^{(j)} = 0$ of each non-isolated gate, the type of $P_k^{(j)}$, called the output type, is defined as the most desirable type of possible cover among those possible covers of $P_k^{(j)}$. (The output type of $P_k^{(j)}$ is COV if it is already covered.)

Generally, the gate type is the same as the least desirable output type of the gate. However, if the gate is isolated, the gate type is ISL (ISoLated); or if the gate is non-isolated, has no input connections, and

all of the outputs are 0 or *, the gate type is LTG (LaTently useful Gate).

The desirability order of the gate types is defined as:

$$\text{ISL, COV, LTG, } G^*, \text{ VC}^*, \text{ GC}^*, G^*C^*, \text{ NWG}$$

(from most to least desirable). Finally, the network type is defined as the least desirable gate type. A network whose type is COV realizes the given function (since all outputs are then covered).

Using these definitions, the variable to be specified is determined as follows:

(1) Locate the gate, k, which defines the network type.

(2) Identify the $P_k^{(j)}$ which defines that gate type.

(3) Determine the free variable associated with the possible cover that defines the output type of that $P_k^{(j)}$.

Of course, it is possible that ties may occur in any of these three steps. If a tie occurs in step (1), the gate with the smallest number is chosen (the gates are numbered 1 to R, with the output gate being number 1). If a tie occurs in step (2) and the gate type is G^*, GC^*, G^*C^*, or NWG, then select the $P_k^{(j)}$ (of those tied) with the smallest j; otherwise, (the type is VC^*) skip step (2). The free variable assignment is made according to the variable selected in step (3), as shown in table form in Fig. 4.2. Rules for breaking ties are included.

In general, the selected free variable is set to 1 and included in the partial solution without being underlined; but when this free variable

is associated with a type NWG cover, the assignment is underlined (for the reason of preventing equivalent permuted solutions).

4.3 Cost Estimation

After selecting and fixing the free variable in AGMT-VAR, an estimate is prepared of the lower bound of the objective function for a completion of the current partial solution (i.e., assuming the least possible cost of completing the current partial solution, the cost of the resulting network is calculated). If this cost exceeds the cost of the current incumbent, it is quite obvious that pursuing the extension of this partial solution would be fruitless. So, in that case, the program immediately backtracks. If the estimated cost bound does not exceed the incumbent cost, the computation proceeds as before; the program again enters CHK-IEQ.

Fig. 4.2 Selection of a free variable to be fixed

type	selected variable	action
COV	none	All inputs are covered; the current solution is a feasible solution.
G^{*}	$\beta_{ik}^{(j)}$	Choose $\beta_{ik}^{(j)}$ that is * with smallest i such that the $P_k^{(j)}$ (of type G^{*}) with the smallest j is covered.
VC^{*}	w_{ik}	Choose w_{ik} that is * with smallest i such that the largest number of uncovered outputs is covered.
GC^{*}	$\beta_{ik}^{(j)}$	Choose $\beta_{ik}^{(j)}$ that is * with smallest i such that the $P_k^{(j)}$ (of type GC^{*}) with the smallest j is covered.
$G^{*}C^{*}$	$\beta_{ik}^{(j)}$	Choose $\beta_{ik}^{(j)}$ that is * with smallest i such that the $P_k^{(j)}$ (of type $G^{*}C^{*}$) with the smallest j is covered.
NWG	$\beta_{ik}^{(j)}$	Choose $\beta_{ik}^{(j)}$ that is * with smallest i such that the $P_k^{(j)}$ (of type NWG) with the smallest j is covered (gate i will be isolated).

5. RESULTS

Let a unique 16 bit binary number be defined for each 4-variable function, f, such that, going from left to right, the bits of the binary number represent the functional values: f (0, 0, 0, 0), f (0, 0, 0, 1), f (0, 0, 1, 0), ..., f (1, 1, 1, 1). The representative functions of the 3,984 P-equivalence classes of 4-variable functions were chosen such that the binary number corresponding to the representative function is less than the binary numbers corresponding to all of the other functions in the class. From these representative functions, 438 were selected* and their respective optimal NOR-gate network realizations were obtained. The optimal networks are listed in Appendix E along with various statistics for each network.

The selected functions are listed in order of their binary numbers which are also rewritten in hexidecimal form for easier reference. Column two of the table shows the minimal number of NOR-gates required to realize each function. These numbers were previously known[12] in all cases except for the two functions which required 8 gates. The fourth column, labeled "cost", is the number of input connections for the optimal networks.

For some functions, two or more distinct optimal networks were obtained. In these cases, only the last network to be generated is listed in the table. Column five, labeled "num. of intercon" shows the number of interconnections for each optimal circuit listed (the number of connections in a given case is easily found by subtracting the entry in this column from the cost).

Where multiple optimal networks are generated, the number of connections, interconnections, or levels of gates may vary (of course, the sum of the number

---

* The method of selection will appear later in this section.

of connections plus the number of interconnections does not vary). The sixth and seventh columns of the table indicate, respectively, the greatest number of levels and the least number of levels found among all optimal networks generated for a given function. Similarly, the eighth and ninth columns show the largest and smallest numbers of interconnections among the optimal networks generated (from this, the largest and smallest numbers of connections are easily deduced).

The next four columns define the pattern of external variable connections for the optimal network listed. The four binary numbers (one in each column) represent the input connections from x_1, x_2, x_3, and x_4, respectively. Within each binary number, i, a 1 in the j-th bit position (counting from left to right) indicates a connection of the external variable, x_i, to the j-th gate of the network (a 0 indicates the absence of such a connection).

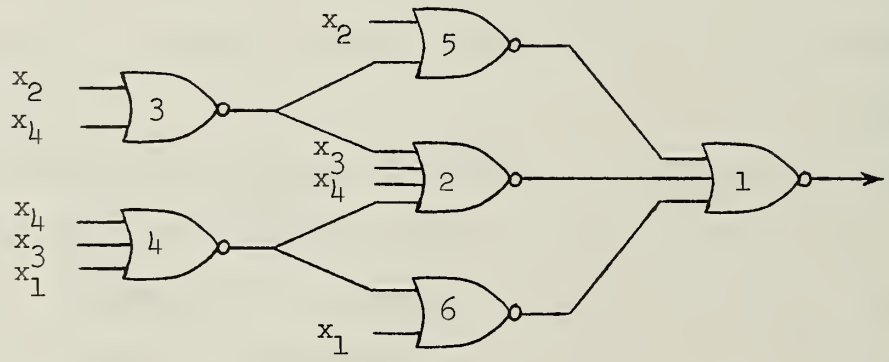
The final column specifies the interconnection of the gates. For each function, the entry consists of several groups of consecutive numbers separated by blanks. If we suppose $\gamma_1\ \gamma_2 \ldots \gamma_m$ is one such group, this is to be interpreted as gate γ_1 feeds gates γ_2, γ_3, ..., γ_m.

For example, consider the function whose binary representation (defined at the beginning of this section) is 1000100010100111. First, convert this number to hexidecimal form*; we get 88A7. Locating the corresponding row in the table, the entries "00010100", "00101000", "01010000", and "01110000" indicate, respectively: x_1 feeds gates 4 and 6; x_2 feeds gates 3 and 5; x_3 feeds gates 2 and 4; and x_4 feeds gates 2, 3, and 4. The gate interconnection pattern is specified by the entries "21 325 426 51 61". This means:

---

* The hexidecimal (base 16) digits A, B, C, D, E, F correspond, respectively, to the base 10 values 10, 11, 12, 13, 14, 15.

gate 2 feeds gate 1; gate 3 feeds gates 2 and 5; gate 4 feeds gates 2 and 6; gate 5 feeds gate 1; and gate 6 feeds gate 1. Then the optimal network is, as constructed from the table listings:



It is interesting to notice that the majority of the optimal networks found for the functions in Appendix E have the same cost as the networks listed by Ikeno et al [12] for these functions.* So, although Ikeno had sought to minimize only the number of gates required to realize each function, in many cases he had also unknowingly found the networks with the minimal number of interconnections plus connections (for that minimal number of gates). The important difference between this work and Ikeno's is that, in this work, the networks are known to be optimal in terms of having the minimal number of connections plus interconnections for the minimal number of gates, while in Ikeno's case only the number of gates is known to be optimal for each function.

---

* Actually, [12] lists only NAND gate networks; but to find the optimal NOR gate network for a given function using this listing, one need only determine the optimal NAND gate network for its dual.

Table E-2 condenses some of the results shown in Table E-1. The number of functions for which optimal networks were obtained is shown for each number of gates. All functions which required 3,4, or 5 gates were exhausted (this fact is known since no other functions were listed by Ikeno as requiring 3, 4, or 5 gates).

The group of 83 functions chosen from the 707 functions of exactly 4 variables requiring (optimally) 6 gates was a fairly biased sample. Of these 83, 32 were chosen because their corresponding P-equivalence classes each had exactly 6 members; the rest were chosen at random.

The 46 7-gate functions were chosen to obtain a good sampling of groups of functions with different numbers of true vectors (i.e., group i consists of all functions of i true vectors, i = 0, 1, 2, ..., 16), as will be shown later in Table C-1 in Appendix C.

Since 8-gate functions require considerable amounts of computation time to determine optimal networks, only two 8-gate functions were selected. And these two were among the 8-gate functions suspected of requiring the least computation times (this will be discussed more later).

For each number of gates, R, the average cost of the optimal networks obtained is shown along with the largest and smallest costs encountered. Also, for each R, the most and least numbers of interconnections and levels are shown. Averages of the columns "most levels", "least levels", "most inter", and "least inter" of Table E-1 for each value of R, complete Table E-2.

5.1 Computation Time Versus Number of Gates

Table 5.1.1 presents the computation times actually required to obtain the optimal networks listed in Appendix E; the times are given as a function of the number of gates. In studying this table, it must be remembered that

the results for 6, 7, and (especially) 8 gates are only partial; only a fraction of the number of functions in these groups were used in obtaining optimal networks.

Table 5.1.1 Computation time vs. number of gates

number of gates	computation time (in sec.)/fn.				number of iterations/fn.		
	least	most	average	total	least	most	average
3	.17	.32	.26	3.44	2	8	3.8
4	.40	.97	.55	33.23	5	23	10
5	.75	4.86	1.35	317.0	9	152	26
6	1.93	32.55	6.36	527.6	33	798	129
7	16.11	435.2	92.25	4244.	336	7666	1700
8	994.7	1514.	1254.	2509.	13455	18948	16202

The results in Table 5.1.1 concerning computation time are graphed in Appendix B. Apparently, the logarithm of the logarithm of the computation time is a linear function of the number of gates (disregarding the 8-gate results as too incomplete to be significant).

5.2 Computation Time Versus Number of True Vectors

Interestingly, within a group of functions requiring the same number of gates, there was found to be a strong correlation between the number of true vectors of a function (i.e., the number of 1's in the binary representation of a function) and the computation time required to exhaust all optimal networks for that function. As the number of true vectors increases, the computation time decreases. Table C-1 in Appendix C presents this correlation in table form.

Perhaps the reason for this correlation is as follows: For every true vector of the output gate, all of the inputs to that gate corresponding to the true vector must be zero; but for every false vector of the output gate, only one of its inputs must be one and the others may be either zero or one, thus the inputs to that gate may assume any of many different combinations of zeros and ones. Therefore, in a rough sense, the more true vectors a function has, the fewer the possibilities which must be tried in synthesizing a network for that function.

As the number of gates increases, the differences in the computation time for different numbers of true vectors become more and more pronounced. The results shown in Table C-1 are given again, this time in the form of a graph, in Fig. C-1.

Notice that this correlation is just a general tendency; any given function may not hold to this pattern. For example, in the 6-gate case we see the computation time rising as the number of true vectors decreases (i.e., as the number of false vectors increases), but the tendency seems to reverse itself as the number of true vectors drops below four. In particular, there is only one function with two true vectors; and, instead of requiring the

longest time, its computation time is even less than the average time for a 6-gate function. So any predictions based on this tendency should only be applied to fairly large samples of functions. Again, remember that the results for 6 or more gates are based on only partial samples.

The two functions which required 8 gates were specifically chosen for their large numbers of true vectors (13 apiece); it was anticipated that only such 8-gate functions would be within computational range. Indeed, this suspicion was supported by the fact that these two functions were among those functions requiring the least amount of time out of a group of 8-gate functions run with a program using a "branch and bound" algorithm.* This is a program specifically tailored to the task of designing NOR networks (and hence, is considerably faster than the implicit enumeration program used for this report), which had proven itself in earlier trials to be a fairly good indicator of the relative difficulty of solving a given function by the implicit enumeration program.

Another interesting observation that we may make at this point is that the "easiest" (requiring the least amounts of computation time) functions of R gates required approximately the same amount of time as the "hardest" (greatest computation times) functions of R-1 gates. The suspected cause is as follows. First, it should be pointed out that no external variables can be connected to the output (NOR) gate of a network realizing a function of 9 or more true vectors (any such connection would force 8 input vectors to be false vectors for this gate). So the program must attempt to connect

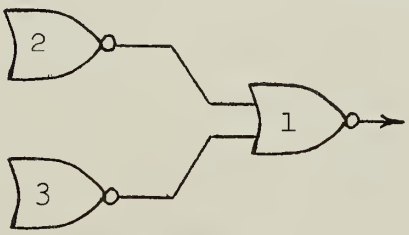
---

* Program developed by T. Nakagawa and H.C. Lai[17] improving on a previous work by Davidson in [6].

other gates to the output gate. One of the cases it considers is the case when only one gate (call it gate number 2) connects to the output gate. What is the output of gate 2? It is just the complement of the original function, and the complement of a function of very many true vectors is a function of very few true vectors (and there are now R-1 gates to realize this second function). So a function of R gates which has a large number of true vectors is somewhat like a function of R-1 gates which has a small number of true vectors. Of course the argument given here is quite crude and ignores many important factors, but it is only intended to point out a likely cause behind the observation.

5.3 Comparison with Three Variable Case

Earlier, C.R. Baugh et al had found optimal networks for 3-variable functions using an all-interconnection NOR formulation.* Since the implicit enumeration program he used was somewhat different from the one used here, the results cannot be considered directly comparable. He assumed unlimited fan-in and fan-out (except for the 8-gate problem, where fan-in, fan-out = 3) and also assumed the three gate subnetwork (gate 1 is the output gate of the network to be synthesized):



* The algorithm used is described briefly in [2] and also the results for the 3-variable parity function (requiring 8 gates) are contained. The results for the other 6- and 7-gate, 3-variable functions discussed in this section are as yet unpublished.

The present program is somewhat improved (various programming gimmicks were added, as well as a subroutine to automatically generate the necessary inequalities) over Baugh's original version, but it does not assume the existence of any such subnetwork.

The average computation times for 3-variable functions are compared with those for 4-variable functions for 6, 7, and 8-gate cases (dotted lines in graph of Appendix B). The average computation times for 3-variable functions of 6 and 7 gates are approximately equal to the lowest computation times for 6- and 7-gate, 4-variable functions (which were investigated for this work), respectively. The computation time for the 3-variable function of 8 gates was far below even the lowest time for a 4-variable function requiring 8 gates.

The ratios between the average computation times for the 3-variable and the 4-variable functions (for the case of 6, 7, and 8 gates) are:

number of gates	ratio (3-var : 4-var)
6	1 : 2.4
7	1 : 5.5
8	1 : 11.4

6. CONCLUSION

This report, combined with earlier reports along similar lines[5],[18], demonstrates the great flexibility of the integer programming approach in the design of optimal networks using different types of gates and network restrictions.

The computations were performed by an IBM 360/75 I system using both the G level and H level Fortran IV compilers (for different subroutines). The program used to solve functions of 7 or fewer gates required 232K bytes of core, while the 8 gate program required about 450K bytes; although the necessary memory space could be reduced by a more ingenious packing of data, computation time would probably increase due to a longer time needed to access this data. Totally, 438 P-equivalence class representatives (representing 6303 functions) were solved in about 7630 seconds.

The results proved that many of Ikeno's[12] networks were optimal both in the number of gates (first) and (secondly) in the total number of connections plus interconnections. For the 13 functions requiring 3 gates, no networks were found better than Ikeno's in terms of our optimality criteria; for the 60 functions requiring 4 gates, 4 networks were improved over Ikeno's. Results were not compared for the cases of more than 5 gates.

It was found that the log of the log of the average computation time apparently varies linearly with the number of gates in the formulation. More importantly, it was discovered that if a group of functions requiring R gates is considered, then the functions with few true vectors tend to take more computation time to determine optimal networks than the functions with many true vectors. Also, the magnitude of this variation seems to increase with R.

Appendix A Backtrack Procedure

The backtrack procedure works, briefly, as follows: A partial solution S is an ordered set of binary assignments to certain variables. An assignment to a variable x_j is said to be underlined if we have already investigated all of the partial solutions, S''s, which have assignments identical to S of the variables preceding x_j, have x_j assigned to the complement (of that value assigned in S), and have arbitrary assignments following. An assignment is not underlined if we have not investigated all such S'.

In the implicit enumeration algorithm we often reach a point where the current partial solution is known to have no feasible completions or a point where it is clear that all possible completions of this partial solution would exceed the cost of the incumbent. We then backtrack to obtain a new partial solution by locating the rightmost non-underlined element of the partial solution (if none exists, the algorithm terminates), setting this element to its complement and underlining, and deleting all elements to its right. This backtrack procedure is also used to obtain the next partial solution after a feasible solution has been found.

Appendix B Number of Gates vs. Computation Time

The graphs in Fig. B-1 and Fig. B-2 show the computation times required to synthesize optimal networks of different numbers of gates. The three solid lines in Fig. B-1 show, for each number of gates: the most, average, and least computation times required to synthesize networks for 4-variable functions needing that number of gates. The dashed line shows the average computation time for some 3-variable functions of 6, 7, and 8 gates.

Fig. B-2, the inset to Fig. B-1, plots the log of the log of the average computation time (in centiseconds) of the 4-variable functions, against the number of gates required to realize those functions.

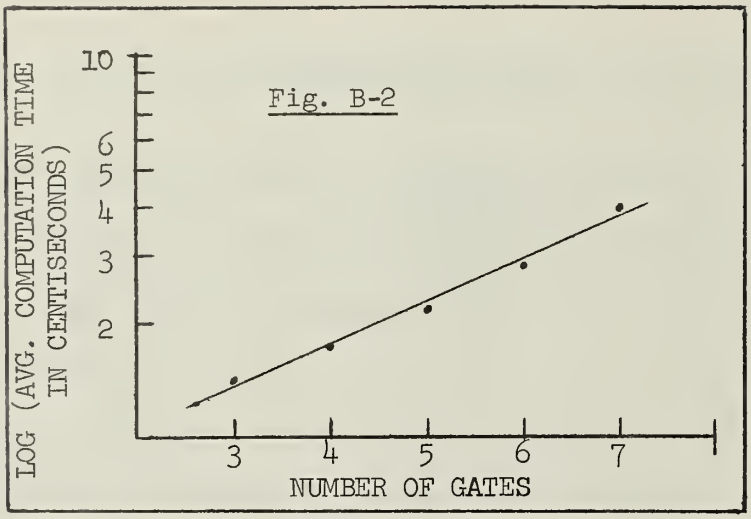


Fig. B-2

Fig. B-1 Computation time vs. number of gates

COMPUTATION TIME (IN SECONDS)

1000

100

10

1

AVERAGE TIME

AVERAGE TIME

MOST TIME

LEAST TIME

4 variable case-solid lines

3 variable case-dashed lines

3 4 5 6 7 8

NUMBER OF GATES

Appendix C Number of True Vectors vs. Computation Time

The first part of this section of the appendix is a table displaying the average computation time, t_{ij}, for each group of functions (for which optimal networks were obtained), F_{ij}, having i true vectors and requiring j gates ($i = 1, \ldots, 15$; $j = 3, \ldots, 8$). The number of elements in each group F_{ij} is also shown.

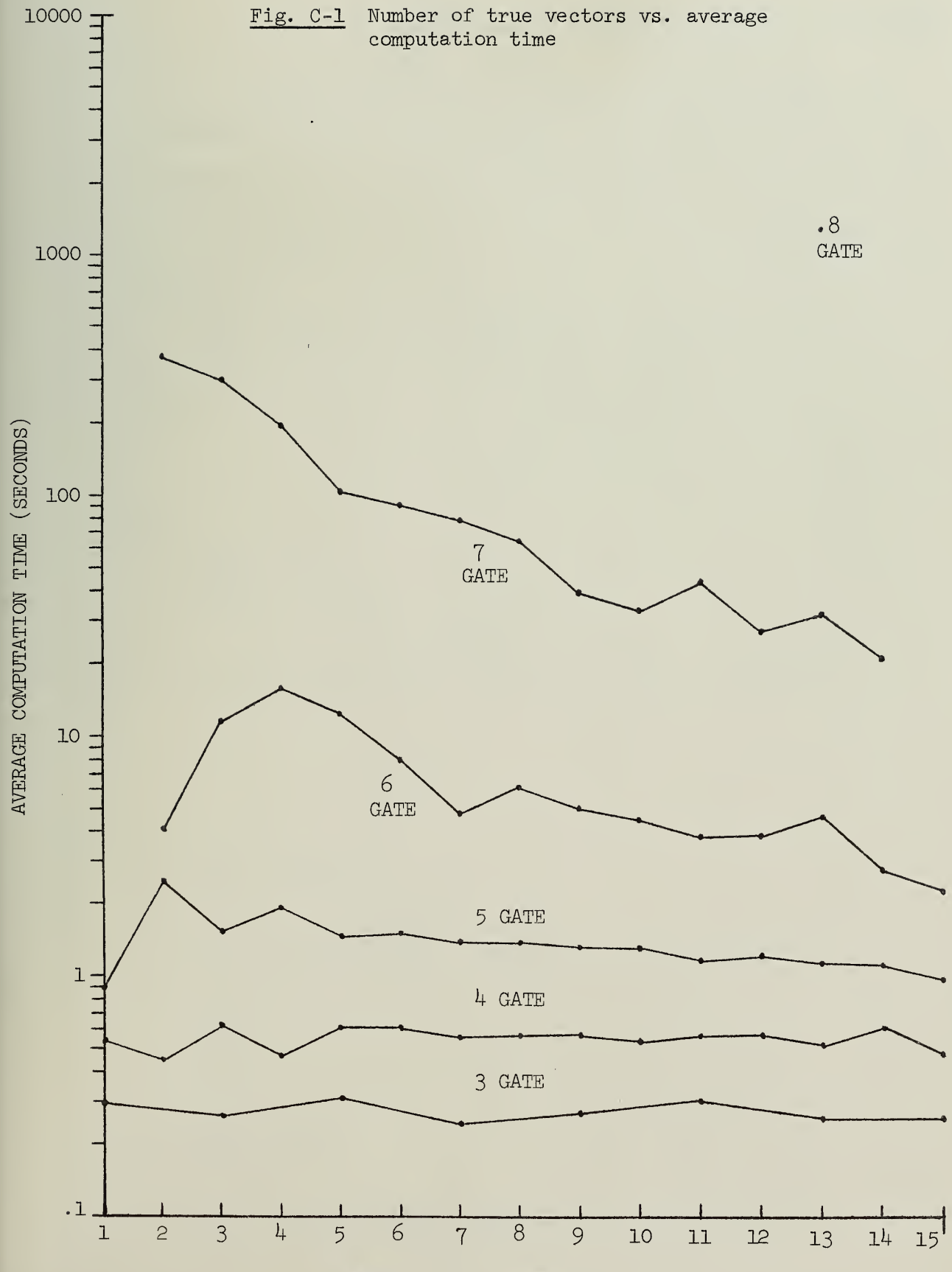
The last part is a graph of the information contained in this table. Notice the slope of the lines from left to right.

Table C-1 Number of true vectors vs. computation time

NUMBER OF TRUE VECTORS	3 GATES		4 GATES		5 GATES		6 GATES		7 GATES		8 GATES	
	NO. OF FN.	AVERAGE TIME PER FN.	NO. OF FN.	AVERAGE TIME PER FN.	NO. OF FN.	AVERAGE TIME PER FN.	NO. OF FN.	AVERAGE TIME PER FN.	NO. OF FN.	AVERAGE TIME PER FN.	NO. OF FN.	AVERAGE TIME PER FN.
1	1	.30	1	.54	1	.89						
2			1	.45	3	2.53	1	4.08	2	376.41		
3	2	.26	4	.63	7	1.53	2	11.63	2	304.53		
4			4	.47	14	1.93	3	15.78	4	197.74		
5	2	.31	7	.61	15	1.45	8	12.35	4	104.54		
6			2	.61	20	1.49	11	8.07	4	91.70		
7	2	.24	6	.56	27	1.37	5	4.68	4	78.95		
8			5	.56	24	1.35	6	6.13	4	63.83		
9	2	.26	7	.56	33	1.30	18	4.95	4	39.00		
10			5	.52	27	1.28	13	4.44	4	32.80		
11	1	.30	7	.55	30	1.16	2	3.77	4	43.01		
12			5	.55	18	1.18	9	3.82	4	27.12		
13	2	.25	4	.51	10	1.12	2	4.44	4	31.39	2	1254.4
14			1	.59	4	1.08	2	2.72	2	20.71		
15	1	.25	1	.47	1	.94	1	2.20				

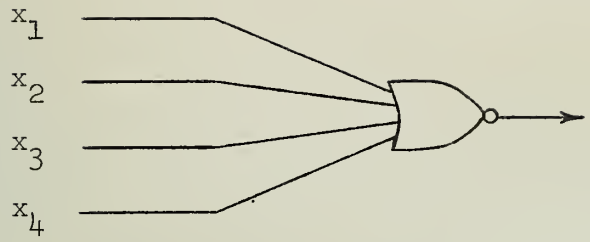
Times are in seconds on the IBM 360/75 I

Fig. C-1 Number of true vectors vs. average computation time

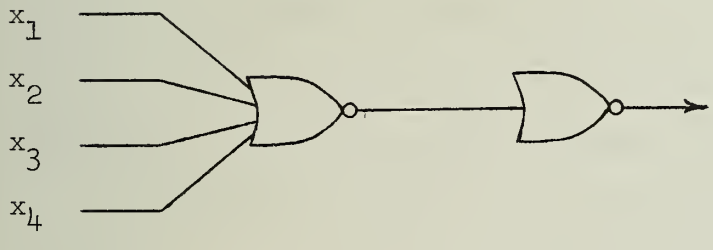


Appendix D Networks of Two or Fewer NOR Gates

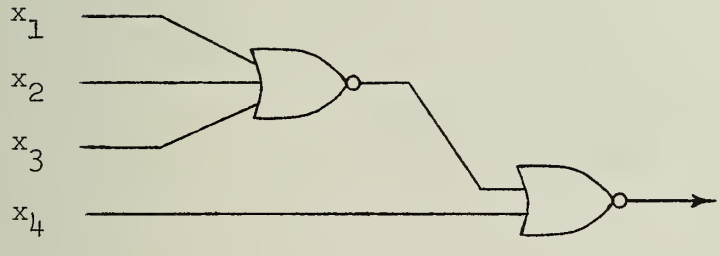
This appendix lists all functions of exactly 4 variables which require fewer than 3 NOR gates. Degenerate functions of 4 variables (i.e., functions which actually depend on only 0, 1, 2, or 3 variables) are not included.



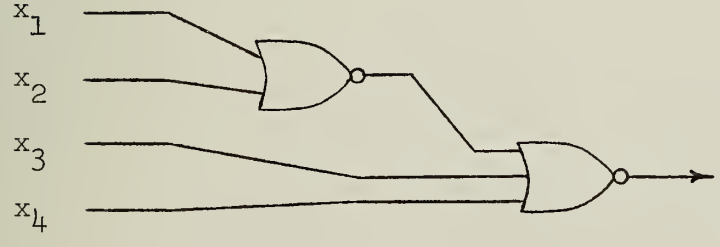
$\bar{x}_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4$



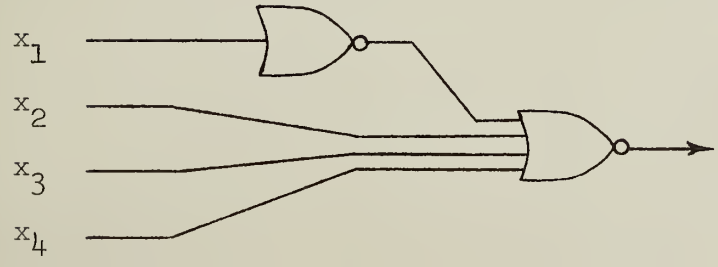
$x_1 \vee x_2 \vee x_3 \vee x_4$



$(x_1 \vee x_2 \vee x_3)\ \bar{x}_4$



$(x_1 \vee x_2)\ \bar{x}_3\ \bar{x}_4$



$x_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4$

Appendix E Optimal Networks

This appendix consists of two tables. The first of these, Table E-1, lists the configurations for all of the optimal networks found in this work. Earlier in the text, it is described how to use this table.

The second table, Table E-2, summarizes some of the information given in Table E-1. The use of this table is also described earlier in the text.

The number of optimal networks for each function was not given due to the difficulty in determining which generated optimal networks are equivalent (i.e., the program may generate the same network twice or more, with a different permutation of gate labels each time).

Table E-1 Optimal Networks
for certain 4-variable
functions

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX1	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
0001	5	0000000000000001	8	4	2	2	4	4	00001000	00010000	00100000	01000000	21 31 41 51
0002	4	0000000000000010	7	3	2	2	3	3	00010000	00100000	01000000	10000000	21 31 41
0007	4	0000000000000111	7	3	2	2	3	3	00010000	00100000	01000000	01000000	21 31 41
0008	3	0000000000001000	6	2	2	2	2	2	00100000	01000000	10000000	10000000	21 31
0009	6	0000000000001001	12	6	3	3	6	6	00001001	00001000	00110000	01100000	21 324 41 51 61
000B	5	0000000000001011	8	4	3	3	4	4	00001000	00010000	01000000	00100000	21 32 41 51
000E	6	0000000000001110	9	5	3	3	5	5	00000100	00001000	00010000	00100000	21 32 42 51 61
0017	5	0000000000010111	11	4	2	2	4	4	00010000	00101000	01001000	01100000	21 31 41 51
0018	7	0000000000011000	13	6	4	4	6	6	00000010	00011000	01001000	01010000	21 32 42 51 64 71
001B	5	0000000000011011	9	4	3	3	4	4	00001000	00010000	01000000	00110000	21 32 41 51
001F	4	0000000000011111	8	3	2	2	3	3	00100000	01010000	00010000	01000000	21 31 41
002A	3	0000000000101010	6	2	2	2	2	2	00100000	01000000	01000000	10000000	21 31
002F	5	0000000000101111	8	4	4	4	4	4	00010000	01000000	00001000	00100000	21 32 41 53
007F	3	0000000001111111	6	2	2	2	2	2	00100000	01000000	01000000	01000000	21 31
0092	5	0000000010010010	11	5	3	3	5	5	00001000	00110000	01100000	10000000	21 324 41 51
0093	5	0000000010010011	11	5	3	3	5	5	00001000	00110000	01100000	00100000	21 324 41 51
0097	5	0000000010010111	12	5	3	3	5	5	00001000	00110000	01100000	01100000	21 324 41 51
009A	4	0000000010011010	7	3	3	3	3	3	00010000	01000000	00100000	10000000	21 32 41
009B	5	0000000010011011	10	5	3	3	5	5	00001000	00010000	01100000	00100000	21 324 41 51
009E	4	0000000010011111	7	3	3	3	3	3	00010000	01000000	00100000	00100000	21 32 41
009F	5	0000000010011111	11	5	3	3	5	5	00001000	00010000	01100000	00110000	21 324 41 51
00BF	5	0000000010111111	12	5	3	3	5	5	00010000	01001000	00101000	01100000	21 324 41 51
00A9	5	0000000011010010	8	4	3	3	4	4	00001000	00010000	00100000	10000000	21 32 42 51
00AB	5	0000000011010011	9	5	3	3	5	5	00001000	00010000	01000000	00100000	21 324 41 51
00BF	4	0000000011011111	7	3	3	3	3	3	00010000	01000000	01000000	00100000	21 32 41
00EA	5	0000000011101010	8	4	3	3	4	4	00001000	00010000	00010000	00100000	21 32 42 51
00EB	6	0000000011101011	13	7	3	3	7	7	00000100	00011000	01010000	00100000	21 325 425 51 61
00EF	5	0000000011101111	9	4	3	3	4	4	00001000	01000000	00010000	00100000	21 32 42 51
00FF	6	0000000011111111	9	5	3	3	5	5	00000100	00001000	00010000	00100000	21 32 42 52 61
011F	6	0000000100011111	15	5	2	2	5	5	00101100	01010100	00011000	01100000	21 31 41 51 61
0129	6	0000000100101011	13	6	4	3	6	6	00000100	00010000	01100000	00101100	21 325 41 54 61
013F	5	0000000100111111	12	4	2	2	4	4	00111000	01001000	01010000	00100000	21 31 41 51
016F	7	0000000101101111	15	8	4	4	8	8	00001000	01000010	00010010	00100010	21 326 426 51 65 71
017F	5	0000000101111111	13	4	2	2	4	4	00111000	01001000	01010000	01100000	21 31 41 51
0180	7	0000000110000000	14	7	4	4	7	6	00011000	00000100	00100010	00101000	21 32 42 51 647 71
0183	5	0000000110000011	12	5	3	3	5	5	00001000	00110000	01100000	00101000	21 324 41 51
0189	7	0000000110001000	13	6	4	4	6	6	00011000	00000010	00100100	00101000	21 32 42 51 64 74
0189	6	0000000110001001	13	7	4	4	7	7	00000100	00001000	00110000	01100000	21 3245 41 51 63
0189	5	0000000110001011	11	5	3	3	5	5	00001000	00010000	01100000	00101000	21 324 41 51
018E	5	0000000110001111	11	4	3	3	4	4	00011000	01000000	00101000	00110000	21 32 41 51
01AB	5	0000000110101011	10	5	3	3	5	5	00001000	00010000	01000000	00101000	21 324 41 51
01AF	5	0000000110101111	10	4	3	3	4	4	00011000	01000000	00001000	00110000	21 32 41 51
01FF	4	0000000111111111	9	3	2	2	3	3	01110000	00010000	00100000	01000000	21 31 41
022A	4	0000001000101010	10	3	2	2	3	3	00110000	01010000	01100000	10000000	21 31 41
0255	6	0000001001010101	12	6	4	4	6	6	00110000	00000100	00001000	01100000	21 324 41 53 63
0257	5	0000001001010111	10	4	3	3	4	4	00010000	00100000	01000000	01101000	21 31 41 54
025F	5	0000001001011111	9	4	4	4	4	4	00100000	01000000	00001000	01010000	21 31 43 54
0297	5	0000001010010111	11	5	4	4	5	5	00001000	00110000	01100000	10000000	21 324 41 53
02A8	6	0000001010001000	11	5	4	4	5	5	00011000	00000100	00101000	10000000	21 32 42 51 64
02BA	4	0000001010101010	8	3	3	3	3	3	00010000	01000000	00100000	10000000	21 32 41

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
02BE	5	0000001011001111	10	5	4	4	5	5	00010000	01000000	00100000	00101000	21 325 41 54
02AA	3	0000001010101010	7	2	2	2	2	2	01100000	00100000	01000000	10000000	21 31
02C4	7	0000001011000100	14	7	5	4	7	7	00011000	00000100	00101000	00010010	21 32 42 51 647 73
02CE	5	0000001011001111	9	5	4	4	5	5	00010000	01000000	00100000	00001000	21 325 41 54
02DE	5	0000001011011111	10	5	4	4	5	5	00010000	01000000	00100000	01001000	21 325 41 54
02FF	5	0000001011111111	8	4	4	4	4	4	01000000	00001000	00010000	00100000	21 32 43 53
0357	5	0000001101010111	12	4	2	2	4	4	00011000	00101000	01010000	01100000	21 31 41 51
035E	4	0000001101011111	9	3	2	2	3	3	00110000	01010000	00100000	01000000	21 31 41
037F	4	0000001101111111	10	3	2	2	3	3	00110000	01010000	01100000	01000000	21 31 41
0383	5	0000001110000011	11	5	4	4	5	5	00001000	00110000	01100000	00100000	21 324 41 53
0387	5	0000001110000111	13	5	3	3	5	5	00001000	00110000	01101000	01100000	21 324 41 51
038B	5	0000001110001011	10	5	4	4	5	5	00001000	00010000	01100000	00100000	21 324 41 53
038E	4	0000001110001111	8	3	3	3	3	3	00010000	01000000	00110000	00100000	21 32 41
038B	5	0000001110010011	9	5	4	4	5	5	00001000	00010000	01000000	00100000	21 324 41 53
03AE	5	0000001110101111	9	4	4	4	4	4	00011000	01000000	00010000	00100000	21 32 41 53
03BE	5	0000001110111111	11	4	3	3	4	4	00011000	01001000	01010000	00100000	21 32 41 51
03C7	5	0000001111000111	12	5	3	3	5	5	00001000	00110000	01101000	01000000	21 324 41 51
03DE	5	0000001111011111	11	4	3	3	4	4	00011000	01001000	00110000	01000000	21 32 41 51
03E0	6	0000001111111100	13	8	4	4	8	8	00001100	00010000	00100000	01000000	21 325 425 526 61
0666	6	0000011001100110	11	5	3	3	5	5	00000100	00000100	00011000	00100000	21 32 42 51 61
0688	7	0000011010000011	18	10	4	4	10	10	00010100	00001000	01110000	00100010	21 32457 426 51 61 76
06AA	6	0000011010101010	13	7	4	3	7	5	00010100	00000100	00100000	00101000	21 325 423 51 61
0777	3	0000011101110111	6	2	2	2	2	2	00100000	00100000	01000000	01000000	21 31
077E	4	0000011101111111	11	3	2	2	3	3	00110000	01010000	01100000	01100000	21 31 41
0787	5	0000011111000111	12	5	4	4	5	5	00001000	00110000	01100000	01100000	21 324 41 53
078E	4	0000011111001111	9	3	3	3	3	3	00010000	01000000	00110000	00110000	21 32 41
07A7	5	0000011111010111	11	5	4	4	5	5	00001000	00110000	01000000	01100000	21 324 41 53
07AE	5	0000011111010111	10	5	4	4	5	4	00001000	01000000	00010000	00110000	21 324 41 53
07B7	5	0000011111011111	13	5	3	3	5	5	00010000	00111000	01001000	01100000	21 325 41 51
07BE	5	0000011111011111	12	4	3	3	4	4	00011000	01001000	01010000	00110000	21 32 41 51
07F7	4	0000011111111011	8	3	3	3	3	3	00010000	00110000	01001000	01000000	21 32 41
07FE	3	0000011111111111	7	2	2	2	2	2	01100000	00100000	01000000	01000000	21 31
0880	5	0000100001000000	10	4	3	3	4	4	00011000	00101000	10000000	10000000	21 32 42 51
0887	6	0000100010000111	18	8	4	4	8	8	00010100	00101000	01110000	01110000	21 3245 426 51 61
088A	5	0000100010001010	11	5	4	3	5	5	00010000	01100000	00101000	10000000	21 325 41 54
088E	5	0000100010001111	11	5	4	3	5	5	00011000	01001000	00100000	00100000	21 324 41 51
08A0	5	0000100010100000	10	4	3	3	4	4	00011000	00101000	00010000	10000000	21 32 42 51
08A1	7	0000100010100001	20	11	4	4	11	11	00010010	00100100	00011000	01110000	21 32456 4257 51 61 71
08A2	5	0000100010100010	12	6	4	4	6	6	00011000	00100000	01010000	10000000	21 324 425 51
08A8	5	0000100010101000	9	4	3	3	4	4	00001000	00011000	00100000	10000000	21 32 42 51
08A9	7	0000100010101001	19	11	4	4	11	11	00000010	00010100	00101000	01110000	21 3257 42356 51 61 71
08AA	4	0000100010101010	7	3	4	4	3	3	01000000	00010000	00100000	10000000	21 32 43
08AE	5	0000100010101111	11	5	4	4	5	5	00010000	01100000	00001000	00101000	21 325 41 54
08B3	5	0000100010110011	13	6	4	4	6	6	00011000	00100000	01010000	00110000	21 324 425 51
08B8	5	0000100010111011	11	5	4	3	5	5	00011000	00001000	01100000	00100000	21 324 41 51
08BF	5	0000100010111111	12	5	4	3	5	5	00010000	01100000	01001000	00101000	21 325 41 54
08F0	5	0000100011110000	10	4	3	3	4	4	00011000	00101000	00010000	00010000	21 32 42 51
08F3	5	0000100011110011	12	6	4	4	6	6	00011000	00100000	01010000	00010000	21 324 425 51
08F7	5	0000100011110111	13	6	4	4	6	6	00011000	00100000	01010000	01010000	21 324 425 51
08F8	5	0000100011111000	9	4	3	3	4	4	00001000	00011000	00100000	00100000	21 32 42 51

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
08F8	5	0000100011111011	11	6	4	4	6	6	00001000	00010000	01100000	00100000	21 325 423 51
08FF	4	0000100011111111	7	3	4	4	3	3	01000000	00010000	00100000	00100000	21 32 43
0991	6	0000100110010001	17	7	3	3	7	7	00010100	00100100	00111000	01100000	21 325 425 51 61
0998	5	0000100110011001	11	5	3	3	5	5	00001000	00001000	00110000	01100000	21 324 41 51
099B	5	0000100110111011	12	5	3	3	5	5	00011000	00001000	01100000	00110000	21 324 41 51
09FF	5	0000100111111111	13	5	3	3	5	5	01011000	00001000	00110000	01100000	21 324 41 51
0AA2	5	0000101010100010	10	4	3	3	4	4	00011000	00101000	01000000	10000000	21 32 42 51
0AB0	7	0000101010110000	13	6	5	3	6	6	00011000	00101000	01000100	01010010	21 32 42 51 61 76
0AB3	5	0000101010110011	12	6	4	4	6	6	00011000	00100000	01000000	00110000	21 324 425 51
0ABB	5	0000101010111011	10	5	4	3	5	5	00011000	00001000	01000000	00100000	21 324 41 51
0ABF	5	0000101010111111	11	5	4	3	5	5	00010000	01100000	01000000	00101000	21 325 41 54
0AC0	5	0000101011000000	10	4	3	3	4	4	00011000	00101000	00100000	00010000	21 32 42 51
0ACF	5	0000101011001111	10	5	4	4	5	5	00010000	01100000	00100000	00001000	21 325 41 54
0ADF	5	0000101011011111	11	5	4	4	5	5	00010000	01100000	00100000	01001000	21 325 41 54
0AE7	7	0000101011100111	18	9	4	4	9	9	00010010	00101000	01000100	01110000	21 3245 427 51 65 71
0AEA	5	0000101011101010	9	4	3	3	4	4	00001000	00011000	00010000	00100000	21 32 42 51
0AF3	5	0000101011110011	11	6	4	4	6	6	00011000	00100000	01000000	00010000	21 324 425 51
0AF7	5	0000101011111011	12	6	4	4	6	6	00011000	00100000	01000000	01010000	21 324 425 51
0AF9	7	0000101011111001	18	10	4	4	10	10	00001010	00010000	00100100	01101000	21 326 4256 527 61 71
0AFB	5	0000101011111011	10	6	4	4	6	6	00001000	00010000	01000000	00100000	21 325 423 51
0BB3	5	0000101110110011	11	4	3	3	4	4	00011000	00101000	01000000	00110000	21 32 42 51
0BB8	4	0000101110111011	7	3	3	3	3	3	00010000	00010000	01000000	00100000	21 32 41
0BBF	5	0000101110111111	12	5	3	3	5	5	00011000	01001000	01010000	00100000	21 324 41 51
0BD1	6	0000101111010001	15	7	3	3	7	7	00010100	00100100	00101000	01010000	21 325 425 51 61
0BDF	5	0000101111011111	13	5	3	3	5	5	00011000	01001000	00110000	01100000	21 324 41 51
0BEE	6	0000101111101111	13	6	4	3	6	6	00001100	01000100	00011000	00100000	21 325 42 51 61
0BF3	5	0000101111110011	10	4	3	3	4	4	00011000	00101000	01000000	00010000	21 32 42 51
0BFB	5	0000101111111011	9	4	3	3	4	4	00001000	00011000	01000000	00100000	21 32 42 51
0BFD	7	0000101111111101	15	9	4	4	9	9	00001100	00010000	00100000	01001000	21 327 427 523 61 76
0BFF	4	0000101111111111	8	3	3	3	3	3	01010000	00010000	01000000	00100000	21 32 41
0EFF	5	0000111011101110	8	4	3	3	4	4	00001000	00001000	00010000	00100000	21 32 42 51
0FEF	6	0000111011101111	13	7	3	3	7	7	00001100	01000100	00010000	00100000	21 325 425 51 61
0FFF	5	0000111011111111	9	4	4	3	5	4	01001000	00001000	00010000	00100000	21 32 42 51
0FF1	6	0000111111110001	13	7	3	3	7	7	00010100	00100100	00001000	01000000	21 325 425 51 61
0FF7	5	0000111111110111	10	4	3	3	4	4	00011000	00101000	01000000	01000000	21 32 42 51
0FF8	6	0000111111111000	11	5	3	3	5	5	00001100	00010100	00100000	00100000	21 32 42 52 61
177F	5	0001011101111111	16	4	2	2	4	4	00111000	01011000	01101000	01110000	21 31 41 51
179F	5	0001011110011111	14	5	3	3	5	5	00010000	01001000	00111000	01110000	21 325 41 51
17AF	6	0001011110101111	14	7	4	4	7	7	00011000	01100000	00000100	00101000	21 324 425 51 64
17BF	5	0001011110111111	13	4	3	3	4	4	00011000	01001000	01010000	00111000	21 32 41 51
17FF	4	0001011111111111	12	3	2	2	3	3	01110000	00110000	01010000	01100000	21 31 41
1988	6	0001100010001000	13	5	4	4	5	5	00011000	00011000	00100100	00100000	21 32 42 51 64
198F	6	0001100010011111	17	8	4	4	8	8	00011000	01001100	00100100	01100000	21 3246 41 534 61
19FF	7	0001100011111110	17	9	4	4	9	9	00001110	00001000	00010010	00100100	21 327 426 524 61 71
1991	6	0001100110010001	18	7	4	3	8	7	00010100	00100100	00111000	01110100	21 325 425 51 61
1999	5	0001100110011001	11	5	4	4	5	5	00001000	00001000	00110000	01100000	21 324 41 53
19B3	5	0001100110110011	14	6	4	4	6	6	00011000	00100000	01010000	00111000	21 324 425 51
19B8	5	0001100110111011	12	5	4	4	5	5	00011000	00001000	01100000	00110000	21 324 41 53
19FF	5	0001100111111111	12	5	4	4	5	5	01010000	00001000	00110000	01100000	21 324 41 53
1AF8	7	0001101011111000	16	8	4	4	8	6	00001010	00010100	00100010	00101000	21 32 426 524 61 74

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEXI	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
1AE0	7	0001101011111101	13	10	4	4	10	9	00011100	00010010	00100000	01011000	21 327 4257 526 61 76
1AB0	7	0011011111011001	13	6	3	3	6	6	00011000	00101000	00000100	00001010	21 32 42 51 61 76
1A81	6	0011011110110001	16	7	4	3	8	7	00010100	00100100	00001000	01110100	21 325 425 51 61
1AB3	5	0001101110110011	12	4	3	3	4	4	00011000	00101000	01000000	00111000	21 32 42 51
1AB8	4	0001101110111011	8	3	3	3	3	3	01010000	00010000	01000000	00110000	21 32 41
1ABF	5	0001101110111111	13	5	3	3	5	5	00011000	01001000	01010000	00101000	21 324 41 51
1ADF	5	0001101111011111	12	5	4	4	5	5	00010000	01100000	00110000	01001000	21 325 41 54
1AF1	6	0001101111110001	15	8	4	4	8	8	00010100	00100100	00001000	01010000	21 325 425 51 634
1AFF	4	0001101111111111	9	3	3	3	3	3	01010000	00010000	01000000	00110000	21 32 41
1EFF	6	0001111011101110	14	8	4	4	8	8	00001100	00001100	00011000	00100000	21 325 425 526 61
1FF8	7	0001111011111011	20	12	4	4	12	12	00001110	00010010	01001000	00100000	21 3246 4267 5246 61 71
1FFF	6	0001111011111111	14	8	4	4	8	8	01000100	00001100	00010000	00100000	21 325 425 526 61
1FF0	6	0001111111110000	13	5	3	3	5	5	00011100	00101100	00000100	00001000	21 32 42 51 61
1FF1	6	0001111111110001	14	8	4	4	8	8	00010100	00100100	00001000	01000000	21 325 425 51 634
1FF7	6	0001111111111011	15	7	4	3	7	5	00010100	00101000	01010000	01000100	21 325 423 51 63
1FFE	7	0001111111111110	15	9	4	4	9	9	00000110	00001010	00010000	00100000	21 326 426 52 627 71
1FFF	3	0001111111111111	8	2	2	2	2	2	01100000	01100000	00100000	01000000	21 31
28B8	5	0010100010001000	11	6	4	3	6	4	00010000	00010000	00101000	10000000	21 325 423 51
28A8	5	0010100010101000	10	4	3	3	4	4	00001000	00011000	00101000	10000000	21 32 42 51
28AA	5	0010100010101010	11	4	4	3	5	4	01001000	00011000	00101000	10000000	21 32 42 51
28AF	5	0010100010101111	13	6	4	4	6	6	00011000	01100000	00001000	00101000	21 324 41 534
28BF	5	0010100010111111	14	6	4	4	6	6	00011000	01001000	01100000	00101000	21 324 41 534
28FF	5	0010100011111111	12	4	4	3	5	4	01001000	00011000	00101000	00110000	21 32 42 51
2900	6	0010100111011101	18	8	4	4	8	8	00011100	00010100	00101000	01110000	21 3245 426 51 61
2AAF	5	0010101010101111	11	5	4	3	5	5	00011000	01001000	00001000	00100000	21 324 41 51
2ABF	5	0010101010111111	11	5	4	4	5	5	00011000	01000000	01000000	00101000	21 32 41 534
2AC0	5	0010101011000000	11	6	4	3	6	4	00011000	00100000	00100000	00010000	21 324 425 51
2ACF	5	0010101011001111	12	6	4	4	6	6	00011000	01100000	00100000	00010000	21 324 425 51
2AD5	5	0010101011010101	12	6	4	4	6	6	00011000	00100000	00100000	01010000	21 324 425 51
2ADF	5	0010101011011111	13	6	4	4	6	6	00011000	01100000	00100000	01010000	21 324 425 51
2AEA	5	0010101011101010	10	6	4	3	6	4	00001000	00010000	00010000	00100000	21 325 423 51
2AEF	5	0010101011101111	11	6	4	4	6	6	00001000	01010000	00010000	00100000	21 325 423 51
2AFF	4	0010101011111111	7	3	4	4	3	3	01000000	00010000	00010000	00100000	21 32 43
2BC0	7	0010101111000000	15	8	6	6	8	8	00011000	00100010	00101000	00000100	21 326 42 51 657 74
2BC3	6	0010101111000011	16	7	3	3	7	7	00010100	00101100	01100100	00010000	21 325 425 51 61
2BCF	5	0010101111001111	13	6	4	4	6	6	00011000	01100000	00101000	00010000	21 324 425 51
2BF5	5	0010101111101111	12	6	4	4	6	6	00011000	01010000	00011000	00100000	21 325 423 51
2BFF	5	0010101111111111	12	5	4	4	5	5	01010000	00010000	01100000	00101000	21 325 41 54
2CC0	5	0010110011001100	11	4	3	3	4	4	00011000	00011000	00101000	00010000	21 32 42 51
2CC3	5	0010110011001101	13	4	3	3	4	4	00011000	00011000	00101000	00010000	21 32 42 51
2CE3	6	0010110011110011	15	8	4	4	8	8	00011000	00100100	01110000	00000100	21 3246 425 51 65
2CFF	5	0010110011111111	11	4	4	3	5	4	01001000	00011000	00101000	00010000	21 32 42 51
2D00	5	0010110100110101	13	6	4	4	6	6	00011000	00011000	00100000	01010000	21 324 425 51
2DF5	5	0010110111101101	13	6	4	4	6	6	00011000	00101000	00010000	01100000	21 325 423 51
2E[illegible]	5	0010111001101101	12	6	4	4	6	6	00001000	00011000	00100000	01010000	21 324 425 51
2EF[illegible]	5	0010111011111111	13	6	4	4	6	6	01001000	00011000	00100000	01010000	21 324 425 51
2F[illegible]	5	0010111100110011	9	4	3	3	4	4	00001000	00001000	00011000	00100000	21 32 42 51
2F[illegible]	5	0010111111111111	10	4	4	3	5	4	01001000	00001000	00011000	00100000	21 32 42 51
2FF1	7	0010111111110001	14	10	4	4	10	10	00010110	00100010	00001100	01010000	21 325 427 51 6245 71
2FF7	5	0010111111110111	14	6	4	4	6	6	00011000	00101000	01100000	01010000	21 324 425 51

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEXI	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
2FFD	6	0010111111111101	14	7	4	4	7	7	00001100	00010100	00100000	01001000	21 325 42 526 61
2FFF	4	0010111111111111	7	3	4	4	3	3	01000000	01000000	00010000	00100000	21 32 43
3DDD	5	0011110111011101	11	6	4	3	6	4	00010000	00010000	00101000	01000000	21 325 423 51
3DFD	5	0011110111111101	10	4	3	3	4	4	00001000	00011000	00101000	01000000	21 32 42 51
3DFF	5	0011110111111111	11	4	3	3	4	4	01001000	00011000	00101000	01000000	21 32 42 51
3EFE	6	0011111011101110	12	7	4	3	7	5	00001000	00001000	00010100	00100000	21 32 426 524 61
6888	6	0110100010001000	17	7	4	3	7	5	00010100	00010100	00101100	00111000	21 325 42 51 623
68AF	6	0110100010101111	19	9	4	4	9	9	00011100	01110000	00010100	00100100	21 325 425 51 6345
6AAA	5	0110101010101010	11	6	4	4	6	6	00010000	00010000	00010000	00101000	21 325 423 51
6AC0	5	0110101011000000	12	6	4	3	6	4	00010000	00101000	00101000	00010000	21 325 423 51
6AD5	6	0110101011010101	17	8	4	3	9	7	01011000	00100100	00100100	01110000	21 3246 425 51 65
6ADF	6	0110101011011111	17	8	4	3	9	7	00011100	01000100	00100100	01010000	21 32 425 51 6345
6AEA	5	0110101011101010	11	6	4	3	6	4	00001000	00010000	00010000	00101000	21 325 423 51
6AFF	5	0110101011111111	12	6	4	3	6	4	01010000	00010000	00010000	00101000	21 325 423 51
6BDF	7	0110101111011111	23	11	4	4	11	11	00011110	01000110	00101100	01010000	21 32 4257 51 63457 71
6BEF	6	0110101111101111	16	8	4	4	8	8	00001000	01110000	00011000	00100100	21 32 425 51 6345
6BFF	6	0110101111111111	17	8	4	4	8	8	01001000	00011000	01110000	00100100	21 3246 425 51 65
6EFE	5	0111011101101110	10	4	3	3	4	4	00001000	00001000	00011000	00101000	21 32 42 51
6EFF	6	0111011101101111	15	7	3	3	7	7	00001100	01000100	00010100	00100100	21 325 425 51 61
6FFF	7	0111011011111110	14	7	4	4	7	6	00000010	00001100	00010100	00100100	21 32 42 526 61 75
6FFF	5	0110111011111111	11	4	4	3	5	4	01001000	00001000	00011000	00101000	21 32 42 51
6FFB	7	0110111111111011	18	9	4	4	9	9	00001100	00010100	01011000	00100010	21 325 427 526 61 76
6FFF	5	0110111111111111	11	5	4	4	5	5	01000000	01000000	00011000	00101000	21 32 42 534
7FEF	6	0111111011101110	13	7	4	3	7	5	00001000	00001000	00010100	00100100	21 32 426 524 61
7FFE	7	0111111111111110	14	6	3	3	6	6	00000110	00001010	00010010	00100010	21 32 42 52 62 71
8002	5	1000000000000010	13	6	3	3	6	6	00101000	00110000	01100000	10000000	21 3245 41 51
8003	5	1000000000000011	13	6	3	3	6	6	00101000	00110000	01100000	00100000	21 3245 41 51
8007	5	1000000000000111	14	6	3	3	6	6	00101000	00110000	01100000	01100000	21 3245 41 51
8008	4	1000000000001000	10	4	3	3	4	4	00110000	01100000	10000000	10000000	21 324 41
800A	4	1000000000001010	10	4	3	3	4	4	00110000	01100000	00100000	10000000	21 324 41
800F	4	1000000000001111	10	4	3	3	4	4	00110000	01100000	00100000	00100000	21 324 41
801F	5	1000000000011111	15	6	3	3	6	6	00110000	01101000	00101000	01100000	21 3245 41 51
8028	7	1000000000101000	15	8	4	4	8	8	00001010	00010100	00100100	10000000	21 32 42 51 6345 72
802A	4	1000000000101010	11	4	3	3	4	4	00110000	01100000	01100000	10000000	21 324 41
802D	7	1000000000101101	18	9	5	5	9	9	00000110	00011000	00100010	01010010	21 324 425 51 61 736
8030	7	1000000000111101	17	7	5	3	9	7	00000110	00011100	00101100	01000100	21 32 42 51 657 71
803F	4	1000000000111111	11	4	3	3	4	4	00110000	01100000	01100000	00100000	21 324 41
807F	4	1000000001111111	12	4	3	3	4	4	00110000	01100000	01100000	01100000	21 324 41
8082	5	1000000010000010	12	6	3	3	6	6	00001000	00110000	01100000	10000000	21 3245 41 51
8083	5	1000000010000011	12	6	3	3	6	6	00001000	00110000	01100000	00100000	21 3245 41 51
8087	5	1000000010000111	13	6	3	3	6	6	00001000	00110000	01100000	01100000	21 3245 41 51
8088	3	1000000010001000	6	2	3	3	2	2	01000000	00100000	10000000	10000000	21 32
808A	4	1000000010001010	9	4	3	3	4	4	00010000	01100000	00100000	10000000	21 324 41
808F	4	1000000010001111	9	4	3	3	4	4	00010000	01100000	00100000	00100000	21 324 41
809F	5	1000000010011111	14	6	3	3	6	6	00010000	01101000	00101000	01100000	21 3245 41 51
80AA	3	1000000010101010	6	2	3	3	2	2	01000000	00100000	00100000	10000000	21 32
80AF	5	1000000010101111	9	4	3	3	4	4	00010000	01001000	00001000	00100000	21 32 41 54
80BF	4	1000000010111111	10	4	3	3	4	4	00010000	01100000	01100000	00100000	21 324 41
80FF	3	1000000011111111	6	2	3	3	2	2	01000000	00100000	00100000	00100000	21 32
8183	5	1000000110000011	13	6	3	3	6	6	00001000	00110000	01100000	00101000	21 3245 41 51

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
81BB	6	1000[illegible]1011	13	6	3	3	6	6	00000100	00011000	01001000	00100100	21 32 41 546 61
81BF	5	1000001011011111	14	6	3	3	6	6	01011000	01100000	00101000	00110000	21 3245 41 51
81EF	6	1000001011101111	15	7	3	3	7	7	00001100	01100000	00010100	00101000	21 326 425 51 61
81FF	5	1000000111111111	15	6	3	3	6	6	01011000	00101000	00110000	01100000	21 3245 41 51
8228	7	1000001000101000	17	10	5	5	10	10	00001100	00010010	00100010	10000000	21 325 425 526 61 734
822A	5	1000001000101010	16	6	3	3	6	6	00111000	01101000	01110000	10000000	21 3245 41 51
822C	7	1000001000101100	18	10	5	5	10	10	00001010	00010100	00100100	00011000	21 325 425 527 634 71
8255	6	1000001010101101	16	8	4	4	8	8	00110100	00000100	00001000	01101000	21 324 41 534 623
8288	6	1000001010001000	12	6	5	4	7	6	00010100	00001000	00100100	10000000	21 32 42 546 61
828A	4	1000001010000010	10	4	3	3	4	4	00010000	01100000	00110000	10000000	21 324 41
82AA	4	1000001010101010	11	4	3	3	4	4	01010000	00110000	01100000	10000000	21 324 41
82C3	6	1000001011000011	12	6	3	3	6	6	00001000	00110000	01100000	00000100	21 324 41 51 65
82FF	6	1000001011101111	15	8	5	4	8	8	00001100	01000100	00010000	00100100	21 325 423 51 635
833F	5	1000001100111111	16	6	3	3	6	6	00111000	01101000	01110000	00100000	21 3245 41 51
835F	5	1000001101011111	16	6	3	3	6	6	00111000	01101000	01110000	01100000	21 3245 41 51
837F	5	1000001101111111	17	6	3	3	6	6	00111000	01101000	01110000	01100000	21 3245 41 51
8387	5	1000001110000111	14	6	3	3	6	6	00001000	00110000	01101000	01100000	21 3245 41 51
838E	4	1000001110101111	10	4	3	3	4	4	00010000	01100000	00110000	00100000	21 324 41
839F	5	1000001110111111	15	6	3	3	6	6	00011000	01101000	01110000	01100000	21 3245 41 51
83C3	6	1000001111000011	12	6	5	5	6	6	00001000	00110000	01100000	00000100	21 324 41 53 65
83C8	7	1000001111001000	16	8	5	5	8	8	00011000	00000110	00101100	00100000	21 325 423 51 62 74
83CF	5	1000001111001111	10	4	3	3	4	4	00010000	01001000	00110000	00001000	21 32 41 54
8305	6	1000001111010101	16	7	3	3	7	7	00011100	00100100	00101000	01100000	21 3256 42 51 61
8307	6	1000001111010111	19	8	3	3	8	8	00001100	00110100	01101000	01110000	21 32456 41 51 61
830F	5	1000001111011111	15	6	3	3	6	6	00011000	01101000	00110000	01100000	21 3245 41 51
83FF	6	1000001111111111	14	6	5	4	8	6	00001100	01000100	00011000	00100100	21 32 42 51 635
83FF	4	1000001111111111	11	4	3	3	4	4	01010000	00110000	01100000	00100000	21 324 41
8644	7	1000011010010101	17	9	5	5	9	9	00010100	00001100	00011000	00100010	21 327 423 546 61 71
8777	4	1000011101110111	12	4	3	3	4	4	00110000	00110000	01100000	01100000	21 324 41
877F	5	1000011101111111	18	6	3	3	6	6	00111000	01101000	01110000	01110000	21 3245 41 51
8783	6	1000011110000011	12	6	4	4	6	6	00010000	00101000	01000100	00000100	21 32 42 51 645
8788	6	1000011110001000	13	8	5	4	8	6	00010100	00001000	00100000	00100000	21 324 426 546 61
878F	4	1000011110001111	11	4	3	3	4	4	00010000	01100000	00110000	00110000	21 324 41
87A7	6	1000011110100111	13	6	5	5	6	6	00000100	00110000	01100000	01001000	21 326 41 54 65
87B7	5	1000011110110111	15	6	3	3	6	6	00010000	00111000	01101000	01100000	21 3245 41 51
87BF	5	1000011110111111	16	6	3	3	6	6	00011000	01101000	01110000	00110000	21 3245 41 51
87E0	7	1000011111100000	16	7	4	4	7	7	00011000	00101010	00000110	00100100	21 32 42 51 645 72
87F7	4	1000011111110111	11	4	3	3	4	4	00010000	00110000	01100000	01100000	21 324 41
87FB	7	1000011111111011	17	8	5	4	9	8	00001110	00010100	01000010	00100010	21 32 42 52 61 7356
87FF	4	1000011111111111	12	4	3	3	4	4	01010000	00110000	01100000	01100000	21 324 41
8880	4	1000100010000000	7	3	3	3	3	3	00010000	00100000	10000000	10000000	21 32 42
8887	6	1000100010000111	17	7	3	3	7	7	00010100	00101000	01110000	01110000	21 325 426 51 61
8889	6	1000100010001001	14	8	3	3	8	8	00000100	00001000	00110000	01100000	21 32456 41 51 61
888A	4	1000100010001010	8	4	3	3	4	4	00010000	01000000	00100000	10000000	21 324 41
8898	5	1000100010001011	11	6	3	3	6	6	00001000	00010000	01100000	00100000	21 3245 41 51
889F	4	1000100010001111	8	4	3	3	4	4	00010000	01000000	00100000	00100000	21 324 41
8898	5	1000100010011011	12	6	3	3	6	6	00001000	00010000	01100000	00110000	21 3245 41 51
889F	5	1000100010011111	13	6	3	3	6	6	00010000	01001000	00101000	01100000	21 3245 41 51
88A0	5	1000100010100000	9	4	4	4	4	4	00011000	00100000	00010000	10000000	21 32 42 53
88A2	5	1000100010100010	11	5	3	3	5	5	00011000	00100000	01010000	10000000	21 32 425 51

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX1	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
88A7	6	1000100010100111	16	7	3	3	7	7	00010100	00101000	01010000	01110000	21 325 426 51 61
8888	5	1000100010101000	8	4	4	4	4	4	00001010	00010000	00100000	10000000	21 32 42 54
88AE	5	1000100010101111	9	4	3	3	4	4	00010000	01000000	00001000	00101000	21 32 41 54
88B3	5	1000100010110011	12	5	3	3	5	5	00011000	00100000	01010000	00110000	21 32 425 51
88BE	4	1000100010111111	9	4	3	3	4	4	00010000	01000000	01100000	00100000	21 324 41
88F0	5	1000100011110000	9	4	4	4	4	4	00011000	00100000	00010000	00010000	21 32 42 53
88F3	5	1000100011110011	11	5	3	3	5	5	00011000	00100000	01010000	00010000	21 32 425 51
88F7	5	1000100011110111	12	5	3	3	5	5	00011000	00100000	01010000	01010000	21 32 425 51
88F8	5	1000100011111000	8	4	4	4	4	4	00001000	00010000	00100000	00100000	21 32 42 54
88FB	5	1000100011111011	10	5	3	3	5	5	00001000	00010000	01100000	00100000	21 325 42 51
8991	6	1000110011001001	18	8	3	3	8	8	00010100	00100100	00111000	01110000	21 325 4256 51 61
8999	5	1000100110011001	12	6	3	3	6	6	00001000	00001000	00110000	01100000	21 3245 41 51
89BB	5	1000100110111011	13	6	3	3	6	6	00011000	00001000	01100000	00110000	21 3245 41 51
89FF	5	1000100111111111	14	6	3	3	6	6	01011000	00001000	00110000	01100000	21 3245 41 51
8AA2	5	1000101010100010	12	5	3	3	5	5	00011000	00101000	01010000	10000000	21 32 425 51
8AAA	3	1000101010101010	6	2	3	3	2	2	01000000	01000000	00100000	10000000	21 32
8ACF	5	1000101011001111	8	4	3	3	4	4	00010000	01000000	00100000	01001000	21 32 41 54
8ADF	5	1000101011011111	9	4	3	3	4	4	00010000	01000000	00100000	01001000	21 32 41 54
8AE3	6	1000101011110011	13	7	5	5	7	7	00011000	00100000	01010000	00000100	21 326 423 51 65
8AFF	5	1000101011111111	8	4	5	5	4	4	01000000	00010000	00001000	00100000	21 32 43 54
8BB3	5	1000101110110011	13	5	3	3	5	5	00011000	00101000	01010000	00110000	21 32 425 51
8BB8	4	1000101110111011	9	4	3	3	4	4	00010000	00010000	01100000	00100000	21 324 41
8BBF	5	1000101110111111	14	6	3	3	6	6	00011000	01001000	01110000	00100000	21 3245 41 51
8BDF	5	1000101111011111	14	6	3	3	6	6	00011000	01001000	00110000	01100000	21 3245 41 51
8BE5	7	1000101111100101	18	9	4	4	9	9	00010010	00101000	00000110	01110000	21 325 4237 51 65 71
8BF3	5	1000101111111011	12	5	3	3	5	5	00011000	00101000	01010000	00010000	21 32 425 51
8BF8	5	1000101111111000	11	5	3	3	5	5	00001000	00011000	01100000	00100000	21 325 42 51
8BFF	4	1000101111111111	10	4	3	3	4	4	01010000	00010000	01100000	00100000	21 324 41
8EE7	6	1000111011101110	11	5	3	3	5	5	00001000	00001000	00010100	00100100	21 32 42 51 65
8EF0	6	1000111111110000	11	5	3	3	5	5	00011000	00101000	00000100	00000100	21 32 42 51 65
8EF7	5	1000111111110111	13	5	3	3	5	5	00011000	00101000	01010000	01010000	21 32 425 51
8EF8	6	1000111111111000	12	6	4	3	6	6	00001100	00010100	00100000	00100000	21 326 42 52 61
8EF9	6	1000111111111001	13	6	3	3	6	6	00001100	00010100	01100000	00100000	21 326 42 52 61
8EFF	3	1000111111111111	6	2	3	3	2	2	01000000	01000000	00100000	00100000	21 32
979F	5	1001011110011111	16	6	3	3	6	6	00010000	01101000	00111000	01110000	21 3245 41 51
97A0	7	1001011110101100	19	11	5	5	11	11	00001010	00010100	00100100	00100010	21 325 425 527 6357 71
97AF	7	1001011110101111	17	8	4	4	8	8	00011000	01100100	00000110	00101000	21 324 425 51 65 74
97BF	5	1001011110111111	17	6	3	3	6	6	00011000	01101000	01110000	00111000	21 3245 41 51
97FF	5	1001011111111111	18	6	3	3	6	6	01011000	00111000	01101000	01110000	21 3245 41 51
9880	7	1001100010000000	15	7	4	4	7	7	00010010	00011000	00001010	00100100	21 32 42 546 61 76
9888	6	1001100010001000	11	6	4	4	6	5	00010000	00010000	00001000	00100100	21 32 42 546 61
9887	7	1001100010110111	19	9	4	4	9	9	00011000	00100010	01000110	01010100	21 325 423 51 657 71
9891	5	1001100110010001	14	6	3	3	6	6	00010000	00100000	00111000	01110000	21 325 425 51
9898	5	1001100110011011	13	6	3	3	6	6	00001000	00010000	01100000	00111000	21 3245 41 51
989F	6	1001100110011111	18	8	3	3	8	8	00010100	01001000	00101100	01110000	21 32456 41 51 61
98B3	5	1001100110110011	13	5	3	3	5	5	00011000	00100000	01010000	00111000	21 32 425 51
98BF	5	1001100110111111	14	6	3	3	6	6	00011000	01000000	01101000	00110000	21 3245 41 51
99F9	6	1001100111111001	13	7	4	4	7	7	00000100	00010000	00101000	01100000	21 325 425 51 64
9AAA	6	1001101010101010	12	7	5	5	7	7	00010000	00010000	00001000	00100100	21 326 423 54 61
9AFF	6	1001101011111111	13	6	5	4	7	6	01000100	00010100	00001000	00100100	21 32 42 546 61

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEXI	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
9983	5	1001110111110011	14	5	3	3	5	5	00011000	00101000	01010000	00111000	21 32 425 51
9988	4	1001101111011011	10	4	3	3	4	4	00010000	00010000	01100000	00110000	21 324 41
998E	5	1001101111011111	15	6	3	3	6	6	00011000	01001000	01110000	00101000	21 3245 41 51
9B0F	4	1001101111011111	10	4	3	3	4	4	00010000	01000000	00110000	01100000	21 324 41
9BE1	7	1001101111111001	17	9	5	5	9	9	00010010	00100010	00001100	01000100	21 325 42 51 657 734
9BE7	6	1001101111111111	18	7	3	3	7	7	00011100	00101000	01010100	01011000	21 32 4256 51 61
9BEB	6	1001101111111011	14	6	5	4	7	6	00001100	00011000	01000100	00101000	21 32 42 51 645
9BEF	4	1001101111111111	11	4	3	3	4	4	01010000	00010000	01100000	00110000	21 324 41
9BF7	6	1001111111111111	19	7	3	3	7	7	00011100	00101100	01010100	01101000	21 326 425 51 61
9BFA	7	1001111111111010	17	8	5	4	9	8	00001110	00010110	00000100	00100010	21 32 42 52 6457 71
9BFB	7	1001111111111011	18	9	5	4	9	8	00001100	00010100	01010010	00100010	21 325 425 526 61 76
9BFF	4	1001111111111111	12	4	3	3	4	4	01010000	01010000	00110000	01100000	21 324 41
A880	5	1010100010000000	11	4	3	3	4	4	00011000	00101000	00110000	10000000	21 32 42 52
A898	4	1010100010001000	7	3	3	3	3	3	00010000	00010000	00100000	10000000	21 32 42
A89A	5	1010100010001010	12	6	3	3	6	6	00011000	01010000	00100000	10000000	21 325 425 51
A8BF	5	1010100011101111	13	6	3	3	6	6	00011000	01010000	00100000	00110000	21 325 425 51
A8AA	4	1010100010101010	7	3	3	3	3	3	01000000	00010000	00100000	10000000	21 32 42
A8AF	5	1010100010101111	11	5	3	3	5	5	00010000	01100000	00001000	00101000	21 324 41 54
A8BF	5	1010100010111111	12	5	3	3	5	5	00010000	01001000	01100000	00101000	21 324 41 54
A8D5	6	1010100011010101	17	7	3	3	7	7	00011100	00100100	00110000	01010100	21 32 425 51 625
A8DD	5	1010100011011101	13	6	3	3	6	6	00011000	00010000	00100000	01110000	21 325 425 51
A8DF	5	1010100011011111	14	6	3	3	6	6	00011000	01010000	00100000	01110000	21 325 425 51
A8EC	6	1010100011111100	9	5	3	3	5	5	00001000	00010000	00100000	00000100	21 32 42 51 65
A8ED	5	1010100011111101	12	6	3	3	6	6	00001000	00010000	00100000	01110000	21 325 425 51
A8EF	4	1010100011111111	8	3	3	3	3	3	01000000	00010000	00100000	00110000	21 32 42
A8F0	6	1010101011010101	15	7	3	3	7	7	00000100	00011000	00100100	01110000	21 326 425 51 61
A8F0	6	1010101011011101	17	7	3	3	7	7	00011100	00010100	00101000	01110100	21 325 426 51 61
A8A8	5	1010101010101010	8	4	3	3	4	4	00001000	00010000	00100000	10000000	21 32 42 52
A8A8	5	1010101010101011	10	6	3	3	6	6	00001000	00010000	01000000	00100000	21 3245 41 51
A8AF	4	1010101010111111	8	4	3	3	4	4	00010000	01000000	01000000	00100000	21 324 41
A8C0	5	1010101011000000	9	4	4	4	4	4	00011000	00100000	00100000	00010000	21 32 42 53
A8CF	5	1010101011001111	10	5	3	3	5	5	00011000	01000000	00100000	00100000	21 32 425 51
A8D5	5	1010101011010101	11	5	3	3	5	5	00011000	00100000	00100000	01010000	21 32 425 51
A8DF	5	1010101011011111	11	5	3	3	5	5	00011000	01000000	00100000	01010000	21 32 425 51
A8E8	5	1010101011101010	9	4	4	4	4	4	00001000	00010000	00010000	00100000	21 32 42 54
A8EF	5	1010101011101111	9	5	3	3	5	5	00001000	01000000	00010000	00100000	21 325 42 51
A8F8	5	1010101011111111	13	6	3	3	6	6	00011000	01001000	01010000	00100000	21 3245 41 51
A8C8	5	1010101011001111	11	5	3	3	5	5	00011000	01000000	00101000	00010000	21 32 425 51
A8FF	5	1010101011111111	10	5	3	3	5	5	00001000	01000000	00011000	00100000	21 325 42 51
AAFF	4	1010101011111111	9	4	3	3	4	4	01010000	00010000	01000000	00100000	21 324 41
ACCC	5	1010110011001100	10	4	4	4	4	4	00011000	00011000	00100000	00010000	21 32 42 53
ACC8	7	1010110011001110	14	8	5	5	8	8	00001100	00001010	00010000	00100000	21 32 42 5467 63 73
ACEC	5	1010110011011001	9	4	4	4	4	4	00001000	00011000	00100000	00010000	21 32 42 53
ACEE	6	1010110011011111	9	4	4	4	4	4	01000000	00011000	00100000	00010000	21 32 42 53
ADDC	5	1010111011011101	12	5	3	3	5	5	00011000	00011000	00100000	01010000	21 32 425 51
ADEA	5	1010111011101101	12	5	3	3	5	5	00011000	00101000	00010000	01100000	21 325 42 51
ADED	5	1010111011101111	11	5	3	3	5	5	00001000	00011000	00100000	01010000	21 32 425 51
ADEF	5	1010111011111111	12	5	3	3	5	5	01001000	00011000	00100000	01010000	21 32 425 51
AEEE	5	1010111011101110	8	4	4	4	4	4	00001000	00001000	00010000	00100000	21 32 42 54
AEEF	5	1010111011111111	8	4	4	4	4	4	01000000	00001000	00010000	00100000	21 32 42 54

FUNCTIONS FOR WHICH OPTIMUM NETWORKS WERE OBTAINED

HEX	GATES	BINARY	COST	NUM. OF INTERCON	MOST LEVELS	LEAST LEVELS	MOST INTER	LEAST INTER	X1 TO	X2 TO	X3 TO	X4 TO	GATE INTERCONNECTIONS
AFF7	5	1010111111110111	12	5	3	3	5	5	00011000	00101000	01000000	01010000	21 32 425 51
AFFF	7	1010111111111110	13	7	3	3	7	7	00000110	00001010	00010000	00100000	21 327 42 52 62 71
B000	5	1011110111011101	13	5	3	3	5	5	00011000	00011000	00101000	01010000	21 32 425 51
B0EE	7	1011110111101111	18	9	4	4	9	9	00001110	01001000	00010010	00100100	21 32 42 526 61 7456
B0F0	5	1011110111111101	12	5	3	3	5	5	00001000	00011000	00101000	01010000	21 32 425 51
B0FF	5	1011110111111111	13	5	3	3	5	5	01001000	00011000	00101000	01010000	21 32 425 51
BFFC	7	1011111101101100	17	9	5	3	9	7	00001010	00010010	00100100	00011000	21 326 42 526 61 723
BFFF	3	1011111111111111	6	2	3	3	2	2	01000000	01000000	01000000	00100000	21 32
E880	6	1110100010000000	17	5	3	3	5	5	00011100	00101100	00110100	00110000	21 32 42 52 62
E888	5	1110100010001000	12	4	3	3	4	4	00011000	00011000	00101000	00110000	21 32 42 52
E88F	6	1110100010001111	18	8	3	3	8	8	00011100	01010100	00100100	00110000	21 325 425 51 625
E8A8	5	1110100010101000	11	4	3	3	4	4	00001000	00011000	00101000	00110000	21 32 42 52
E8FF	5	1110100011111111	11	4	3	3	4	4	01000000	00011000	00101000	00110000	21 32 42 52
EAAA	4	1110101010101010	7	3	3	3	3	3	00100000	00100000	00100000	00100000	21 32 42
EAA0	6	1110101010101011	16	9	3	3	9	9	00010100	00011000	01010000	00100000	21 3256 4256 51 61
EAAF	5	1110101010101111	12	6	3	3	6	6	00011000	01010000	00100000	00100000	21 325 425 51
EABF	5	1110101010111111	13	6	3	3	6	6	00011000	01010000	01010000	00100000	21 325 425 51
EAC0	4	1110101011000000	7	3	3	3	3	3	00010000	00100000	00100000	00100000	21 32 42
EAC8	5	1110101011001000	10	4	3	3	4	4	00010000	00001000	00101000	00110000	21 32 42 52
EACF	5	1110101011001111	12	6	3	3	6	6	00011000	01100000	00100000	00010000	21 325 425 51
EAD5	5	1110101011010101	12	6	3	3	6	6	00011000	00100000	00100000	01010000	21 325 425 51
EADF	5	1110101011011111	13	6	3	3	6	6	00011000	01100000	00100000	01010000	21 325 425 51
EAEF	5	1110101011101111	11	6	3	3	6	6	00001000	01010000	00010000	00100000	21 325 425 51
EAFF	4	1110101011111111	7	3	3	3	3	3	01000000	00010000	00010000	00100000	21 32 42
EBEF	5	1110101111101111	12	6	3	3	6	6	00001000	01010000	00011000	00100000	21 325 425 51
EBFE	8	1110101111111110	18	11	5	4	11	10	00000101	00001010	00010001	00100000	21 326 42 527 625 71 846
EBFF	5	1110101111111111	13	6	3	3	6	6	01001000	00011000	01010000	00100000	21 325 425 51
EFEA	5	1110111011101010	9	4	3	3	4	4	00001000	00010000	00011000	00100000	21 32 42 52
EFEF	5	1110111011101111	10	6	3	3	6	6	00001000	01000000	00010000	00100000	21 325 425 51
EFF7	5	1110111111110111	14	6	3	3	6	6	00011000	00101000	01010000	01100000	21 325 425 51
EFF9	8	1110111111111001	19	11	3	3	11	11	00001001	00010001	00100100	01000010	21 328 426 526 61 768 81
EFFA	6	1110111111111011	14	7	3	3	7	7	00001100	00010100	01001000	00100000	21 326 42 526 61
EFFE	7	1110111111111110	13	7	4	4	7	7	00000110	00001010	00010000	00100000	21 32 42 52 62 756
EFFF	4	1110111111111111	7	3	3	3	3	3	01000000	01000000	00010000	00100000	21 32 42
FEFC	5	1111111011101110	8	4	3	3	4	4	00001000	00001000	00010000	00100000	21 32 42 52
FEFF	6	1111111011101111	14	8	3	3	8	8	00001100	01001000	00010000	00100000	21 326 426 526 61
FFFF	5	1111111011111111	8	4	3	3	4	4	01000000	00001000	00010000	00100000	21 32 42 52
FFFE	6	1111111111111110	9	5	3	3	5	5	00000100	00001000	00010000	00100000	21 32 42 52 62

Table E-2

STATISTICS ON OBTAINED OPTIMAL NETWORKS

# OF GATES	# OF FNS	AVG COST	LRGST COST	SMLST COST	MOST INTR	LEAS INTR	AVG MOST INTR	AVG LEAS INTR	MOST LEV	LEAS LEV	AVG MOST LEV	AVG LEAS LEV
3	13	6.308	8	6	2	2	2.000	2.000	3	2	2.462	2.462
4	60	8.967	12	7	4	3	3.467	3.467	4	2	2.917	2.917
5	234	11.479	18	8	6	4	5.064	4.974	5	2	3.321	3.235
6	83	14.277	19	9	9	5	6.916	6.627	5	2	3.639	3.422
7	46	16.478	23	13	12	6	8.587	8.326	6	3	4.304	4.087
8	2	18.500	19	18	11	10	11.000	10.500	5	3	4.000	3.500

LIST OF REFERENCES

[1] Balas, E., "An Additive Algorithm for Solving Linear Programs With Zero-One Variables," Operations Research, vol. 13, no. 4, pp. 517-549, July-Aug., 1965.

[2] Baugh, C.R., T. Ibaraki, T.K. Liu, and S. Muroga, "Optimal Network Design Using NOR and NOR-AND Gates by Integer Programming," Report no. 293, Department of Computer Science, University of Illinois, January, 1969.

[3] Breuer, M.A., "Implementation of Threshold Nets by Integer Linear Programming," IEEETEC, vol. EC-14, no. 6, pp. 950-952, Dec., 1965.

[4] Cameron, S.H., "The Generation of Minimal Threshold Nets by an Integer Program," IEEETEC, vol. EC-13, no. 3, pp. 299-302, June, 1964.

[5] Chandersekaran, C.S., "Synthesis of Optimal Double-Rail Logic Networks Using NOR-OR Gates by Integer Programming," Report no. 384, Department of Computer Science, University of Illinois, February, 1970.

[6] Davidson, E.S., "An Algorithm for NAND Decomposition of Combinational Switching Systems," Ph.D. dissertation, Department of Electrical Engineering and Coordinated Science Laboratory, University of Illinois, 1968.

[7] Fleischman, B., "Computational Experience With the Algorithm of Balas," Operations Research, vol. 15, no. 1, pp. 153-155, Jan.-Feb., 1967.

[8] Geoffrion, A.M., "Integer Programming by Implicit Enumeration and Balas' Method," SIAM Review, vol. 9, no. 2, pp. 178-190, April, 1967.

[9] Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, vol. 13, no. 6, pp. 879-919, Nov.-Dec., 1965.

[10] Hellerman, L., "A Catalog of Three-Variable OR-Invert and AND-Invert Logical Circuits," IEEETEC, vol. EC-12, no. 3, pp. 198-223, June, 1963.

[11] Ibaraki, T., T.K. Liu, C.R. Baugh, and S. Muroga, "Implicit Enumeration Program for Zero-One Integer Programming," Report no. 305, Department of Computer Science, University of Illinois, January, 1969.

[12] Ikeno, N., A. Hashimoto, and K. Naito, "A Table of Four-Variable Minimal NAND Circuits," Electrical Communication Lab. Tech. J., Extra Issue no. 26, Electrical Communication Laboratory, Nippon Telegraph and Telephone Public Corporation, Tokyo, Japan, 1968 (in Japanese).

[13] Liu, T.K., "A Code for Zero-One Integer Linear Programming by Implicit Enumeration," Master thesis, Department of Computer Science, University of Illinois, 1968.

[14] Muroga, S. and T. Ibaraki, "Logical Design of an Optimum Network by Integer Linear Programming-Part 1," Report no. 264, Department of Computer Science, University of Illinois, July, 1968.

[15] Muroga, S. and T. Ibaraki, "Logical Design of an Optimum Network by Integer Linear Programming-Part 2," Report no. 289, Department of Computer Science, University of Illinois, December, 1968.

[16] Muroga, S., "Logical Design of Optimal Digital Networks by Integer Programming," in Advances in Information Systems Science, vol. 3, J.T. Tou, Ed. Plenum Press: New York, 1970, pp. 283-348.

[17] Nakagawa, T., and H.C. Lai, "A Branch-and-Bound Algorithm for Optimal NOR Networks," Report no. 438, Department of Computer Science, University of Illinois, 1971.

[18] Swee, R.S., "Optimum Network Design Using NOR-OR Gates by Integer Programming," Master thesis, Department of Computer Science, University of Illinois, 1970.

[19] Taniguchi, K., N. Tokura, T. Kasami and H. Ozaki, "Logical Functions Realizable by a Planar NAND Network," The Transactions of Electronics and Communications Engineers of Japan, vol. 51-C, no. 2, pp. 59-65, February, 1968.